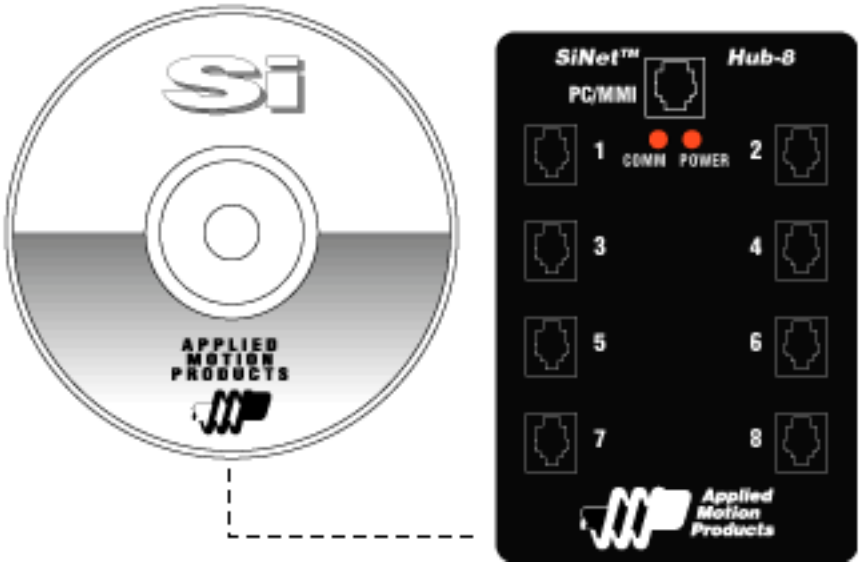
9/18/02

SiNet™ Hub Programmer

Software Manual



Applied Motion Products

Copyright 2000

Contents

Getting Started 5
Installing the Programming Software 6
Display Settings 6
Connecting to your PC 7
Applying Power 7
Programming 8
Introduction 8
Idle Current Reduction 8
Which Software Version do I Have? 9
Command Buttons 10
Download, Upload & Execute 10
Save, Open, Print & Quit 11
Defining Your Hardware 12
Reaction to a Limit 13
Entering Your Program 14
Animation 16
Editing Tips 16
The Instructions 17
MMI Prompt 18
How to display a message on the MMI 19
How to pause until the user presses ENTER 19
How to let the user make a decision (MMI branching) 20
How to ask the user for a move distance 20
How to get a speed from the user 21
How to get a repeat count from the user 21
How to create an MMI Menu 22
What is the String Pool? 22
Feed to Length 23
Feed & Set Output 25
Feed & Return 26
Feed to Sensor 27
Feed to Sensor & Return 29
Feed to Position 30
Set Abs Position 31
Save Abs Position 32
Seek Home 33
Start / Stop Motion 34
Wait Time 35
Wait Input 36
Go To 37
Repeat/End Repeat 38
Reset Repeat Loop 40
Set Output 41
If Input Go To 42
If Moving 44
Change Current 45
Comment 46
Call and Return 47

Getting Started

Thank you for purchasing the Applied Motion Products SiNet™ Hub, the industry's first motion control network hub. We hope you will find that the performance, price and ease of programming make our products the best value for your application.

The SiNet™ Hub Programmer software allows you to quickly and easily program your hub for the control of up to 8 Si™ drives (4 drives with the Hub 444). Not only can you control all the motors, the hub also has access to the inputs and outputs of all the drives, providing your program with up to 64 inputs and 24 outputs. (The Hub 444 also includes on-board four programmable inputs and four programmable outputs.) You can also use our MMI-01 man machine interface terminal to interact with a machine operator.

There are many Si™ drives that you can use with the hub, allowing you to optimize the motor type and power evel of each axis. This manual explains how to install the *SiNet Hub Programmer™* Windows application and how to program your SiNet™ hub.

For information regarding your specific Si™ hardware, such as wiring and mounting, please read the hardware manual that came with that product.

The *SiNet Hub Programmer™* features include:
- Powerful, flexible, easy to use indexer.
- Nonvolatile program storage, up to 200 program lines, 50 MMI variables and 1500 MMI text characters.
- Automatic, stand alone execution of stored program.
- Connection by a simple cable to your PC for programming (cable included).
- Microsoft Windows-based software for easy set up and programming (requires Windows 95, 98 or NT)
- Programmable inputs for interacting with the user and other equipment (8 inputs per drive).
- Programmable outputs for coordinating external equipment (3 outputs per drive).
- Instructions for motion, triggering, branching, loops, time delays and more.
- Ability to work in user defined units such as inches, degrees, gallons, etc.
- Optional man machine interface (MMI) allows operator to enter distances, speeds, loop counts and more.

To operate your SiNet™ Hub, you must do the following:
- Connect at least one Si™ drive.
- Connect a motor to your drive (for the Si-100, you'll need a motor and a pulse & direction drive).
- Connect power to the drive(s).
- Connect any inputs or outputs that you require.
- Plug into your personal computer for programming.
- Install our software program on your PC.
- Have Fun!

Remember, if you have trouble getting your SiNet™ Hub to meet your expectations, or if you want to suggest improvements to the product or this manual, give us a call at (800) 525-1609. Or, you can fax us at (831) 761-6544. If you are not calling from North America, dial (831) 761-6555.

Note: Many of the Si™ drive models that work with this hub are the same popular drives that we've been manufacturing and selling for years. However, to make them work with the Hub and the multi-axis motion control programs that you're about to create, the drives need firmware version 1.75 or later. Call the distributor in your area if you need to upgrade the firmware in your drives.

Installing the Programming Software

The *SiNet Hub Programmer™* software comes on a CD. Before you can use the software, you must install it onto your hard drive. If your computer doesn't have a CDROM drive, you can call the factory, at 831-761-6555 and ask for a set of diskettes. You can also download the files for the three disks from our website: www.applied-motion.com. The diskette files are also included on the CD so that you can use another PC to make installation diskettes.

To run the *SiNet Hub Programmer™* software, you must have a computer with the following requirements:
- IBM compatible 486, Pentium or better CPU. Pentium recommended for best performance.
- Microsoft Windows 95, Windows 98, or Windows NT. *This software does not work with Windows 3.1.*
- At least 8 MB memory (16 MB or more will make the software run much faster)
- 10 MB available hard drive space
- VGA monitor, or better. 16 bit color setting recommended (65,535 colors, sometimes called High Color)
- Mouse or other input device
- CDROM drive. (You can use a 3.5" floppy disk drive instead if you request the optional diskettes.)
- A nine pin serial port must be available, preferably COM1.

The software installation is highly automated, like most Windows programs, so the process is simple:
- Put the CD into your CDROM drive and wait.
- After a few seconds, the installation program will start automatically, and guide you through the rest of the installation.

- The rest of this manual will be installed to the same hard disk folder as the software.
- To read the rest of this manual, you need Adobe Acrobat Reader, which will also be installed automatically.
- The hardware manuals for your hub and Si drives will also be automatically installed.
- The Si Programmer and SCL Setup Utility will also be installed.
- **By choosing Custom Setup, you can choose exactly which software program and manuals are installed.**

If you encounter errors during installation, it is usually due to lack of memory or conflicts with other programs that are already running on your PC. If you experience an error while installing the programming software, quit all other Windows applications and try again. Holding down the ALT key and pressing TAB will show you all the programs currently running on your PC. Laptop computers generally present the biggest challenge to installation, as they often come preloaded with programs that automatically execute on start-up such as Microsoft Office and battery managers. Furthermore, laptops usually have the least memory.

The programming software will install more easily and run much faster if you have more memory. We recommend at least 16 megabytes of RAM.

Several example programs are installed with your programming software. It's a good idea to load some of the examples and look at them; they may help you with your own application.

Display Settings

The *SiNet Hub Programmer™* works well with any display resolution. At 640 x 480, the *SiNet Hub Programmer* will exactly fill your screen. At higher resolutions, like 800 x 600 or 1024 x 768, there will be room left over on the screen for other applications, or to expand the *SiNet Hub Programmer* window so you can see more program lines. We recommend a display color setting of 65,535 colors (sometimes called 16 bits or High Color).

Information in the program window will not display correctly if your display is set for "Large Fonts."
Please use the "Small Fonts" setting when running the *SiNet Hub Programmer* software. The display settings are found under "Start…Control Panels".

Programming Note: Always apply power to Si hardware after the SiNet Hub Programmer™ software is running on your PC.

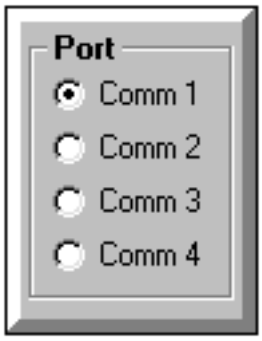
Connecting to your PC

• Locate your computer within 6 feet of the SiNet™ hub. The hub can be farther away (up to 50 feet) if you replace the cable that we supply with a longer one.

• The Si drives that you purchased for use with your SiNet™ hub come with a black adapter plug. It has a telephone style jack at one end and a larger 9 pin connector at the other. Plug the large end into the COM1 serial port of your PC. Secure the adapter with the screws on the sides. If the COM1 port on your PC is already used by something else, you may use the COM2 port for the Si™ Indexer. On some PCs, COM2 will have a 25 pin connector that does not fit the black adapter plug. If this is the case, and you must use COM2, you will have to purchase a 25 to 9 pin serial adapter at your local computer store.

• Your SiNet™ Hub was shipped with a 7 foot telephone line cord. Plug one end into the adapter we just attached to your PC, and the other end into the PC/MMI jack on your SiNet™ Hub.

Never connect any Si™ device to a telephone circuit. They use the same connectors and cords as telephones and modems, but the voltages are not compatible.



You may also need to set the COM port in the Windows software. When the software is loaded, it looks for an available COM port, but doesn't always find the one you've plugged into.

You can choose the port yourself by clicking on one of the "COM port" option buttons. If the port exists and is not already in use, the programming software will use it to communicate with the Si™ Indexer.

Applying Power

When the SiNet Hub wakes up, it automatically checks to see if the *SiNet Programmer* software is running on your PC. If it finds it, the firmware version number is displayed on the main screen, and the hub enters "development mode." When you're done programming the hub, you'll probably want it to wake up and run your program automatically. That's easy: just turn off the hub and disconnect it from your PC. The next time power is applied, the hub will run your program on it's own.

Thus, when you want to operate the hub in development mode, you must connect it to the PC and launch the SiNet Hub Programmer software before applying power to the hub.

In addition to the hub detecting your PC, the drives must detect the hub. That's because they too can run stand alone, in a single axis mode. In fact, there's quite a few things that the Si drives can do based on what they are connected to when power is applied.

If you want your Si drives to work reliably with the SiNet hub, power must first be applied to drive 1. The other drives can be powered up at the same time as drive 1, which is what most people do. But if you have a large number of drives on one system, the combined power up surge maybe be too much for your power system to handle. You could trip a breaker or blow a fuse.

If you want to power up the drives in an orderly sequence, apply power to drive 1 first. Then apply power to the other drives within 1/2 second.

Programming

Introduction

Building a motion control program for your SiNet Hub consists of three parts. First, you need to tell the hub some things about the drives and motors you are using. This information is summarized on the left side of the screen, under the heading “Drives.” Clicking on the “Change...” button brings up a dialog box that lets you define which drive models you have, how much current each motor needs, and more. That’s all covered in detail in the section “Defining Your Hardware.”

Next, you’ll need to tell the hub what to do if one of the drives hits a limit sensor. Refer to the “Limit Sensors” sections for more information about that.

Finally, you get to the really fun part: building the program steps. That takes place on the right hand part of the screen. Your program consists of a series of “lines”. Each line defines a specific event that you want to take place, like moving a motor, activating a programmable output, or asking the operator to enter information on a terminal.

You define these events by putting instructions on each line of your program. There are 23 different instructions to choose from on each of the 200 lines. Of course, you don’t have to use all the instructions or all 200 lines. Many users find that the instructions are so powerful that their program only requires a few lines.

Since the primary mission of the SiNet Hub is motion control, you won’t be surprised to learn that 7 of the instructions make the motor move. You can choose from fixed distance moves like Feed to Length or Feed and Return. You can make the motor run until the load trips a sensor. Feed to Sensor, Feed to Sensor & Return and Seek Home are used for that. Feed to Position moves to an absolute position. Feed and Set Output allows you to move the motor a fixed distance, and activate an output during the move.

Other instructions control the flow of your program, helping to determine the order in which instructions take place. Some examples are Go To, If Input Go To, Repeat, Call and Return.

There is one very powerful, yet easy to use instruction that lets an operator interact with your program while it’s running: the MMI Prompt. If you purchase our optional MMI-01 terminal, you can set up a program that will allow a machine operator to set move distances and speeds, to make yes or no decisions effecting program flow, or to choose program options from a menu. The MMI Prompt also lets you display messages telling the operator what the system is doing.

Wait Input gives you the ability to have your program wait for an outside event, and Set Output lets your program turn other devices on or off, or send signals to another intelligent device like a PLC.

Idle Current Reduction

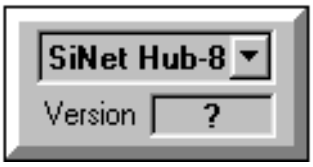
All Si drives are equipped with a feature that automatically reduces the motor current anytime the motor is not moving. This reduces motor and drive heating. The idle current of any Si drive used with the SiNet hub is 50% of the current you set in the Configure Drives dialog.

If you need different idle current values, you can use Change Current instructions in your program to change the current of any drive, anytime.

The Si-100 indexer doesn’t have a drive built into it. So the current, idle current and step resolution of any axis that uses an Si-100 will be controlled by the drive that’s connected to the Si-100.

Which Software Version do I Have?

There are actually two software programs associated with the SiNet™ Hub. The first is the *SiNet Hub Programmer™* Windows program that you installed on your PC. After you double click on the icon and the program begins to load, you'll see a picture of an ocean wave. (In case you're interested, we chose the wave for our "splash screen" for several reasons. Nothing "splashes" quite like a wave, and waves invoke images of motion. Also, Applied Motion is located on the California coast.)

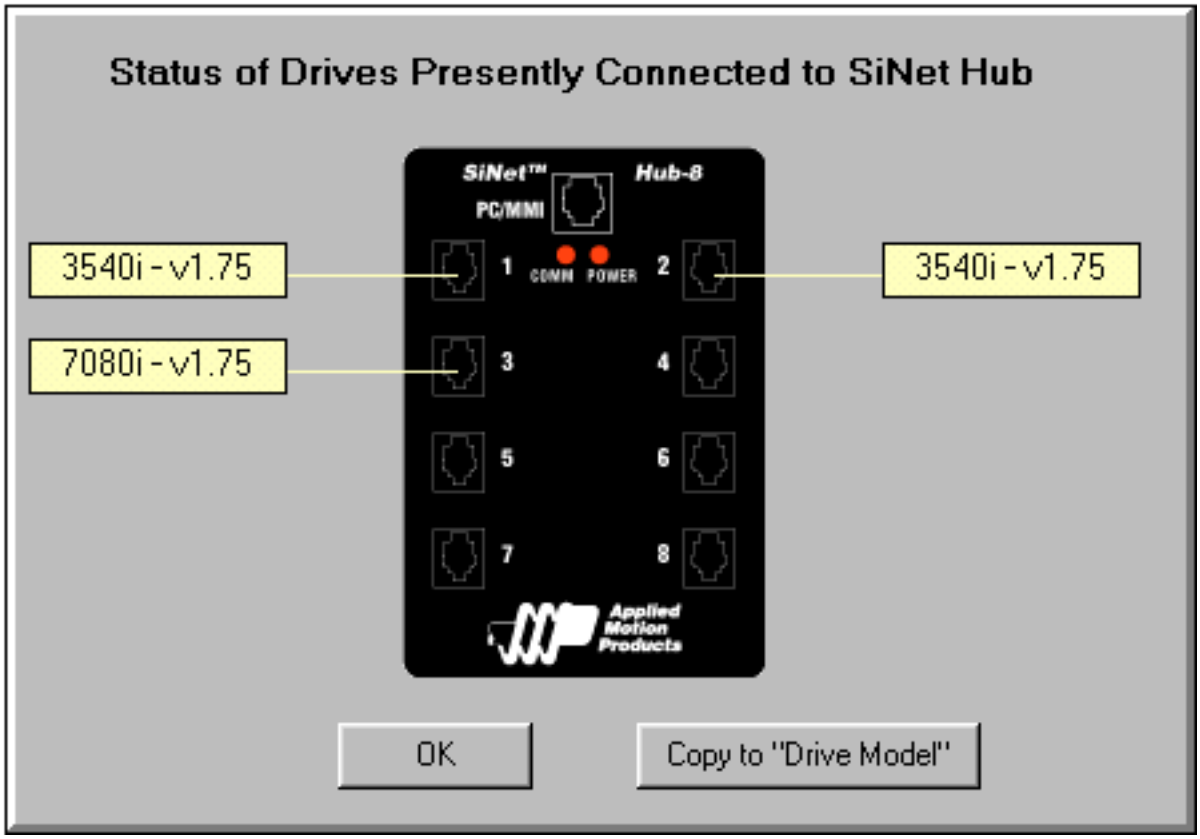


The software version is displayed with the splash screen. After the program is loaded, you can click on the Applied Motion Products logo to see the software version, along with our phone and fax numbers, our internet address and information about the drive products that can be used with the hub.

A second software program resides in a chip inside the SiNet™ Hub. Since software in a chip is usually called "firmware", we will refer to it as firmware for the rest of this manual. It is the SiNet™ Hub firmware that executes the program you've downloaded. When you connect your hub to the PC and turn on drive #1, the hub firmware version is displayed near the top of the screen.

If you call your distributor or Applied Motion Products for support, we will want to know the versions of both the *SiNet Hub Programmer™* and the hub firmware, so please write them down before calling.

There are also firmware programs in each of the Si drives that are used with the hub. If you want to check the firmware version of your drives, connect the hub to the PC, connect the drives to the hub, and power everything up at once. Then click on the "Change..." button under the Drive information display. You will see the Hardware Setup dialog. Near the bottom of the hardware dialog is a button marked Scan Drives. Clicking on this button will show you a picture of the hub, with information about each of the drives that are connected to it, as shown below.



Command Buttons

Download, Upload & Execute

The SiNet™ Hub was designed to operate without a host computer once your program is finished and tested. At first, though, you'll probably find yourself running it from the PC much of the time. That way you can quickly make changes in your program to fix errors or conduct experiments.

The *SiNet Hub Programmer™* software provides four command buttons for interacting with the indexer-drive.

Download transfers the program from the Windows software to the SiNet™ Hub that's plugged into your serial port. The transfer takes a few seconds. You must download the program before you can execute it. If you press the Execute button before downloading, the program that was previously in your hub will execute and the results may not match what is shown on your screen.

Upload lets you extract whatever program is in the SiNet™ Hub and display it on the screen. If you want to modify a program already in your SiNet™ Hub, you can use the Upload command to bring it back to the PC.

Execute tells the SiNet™ Hub to begin running the program that is in its internal memory, starting on line 1. After hitting the Execute button, you'll see a box appear with a status display and five program control buttons.

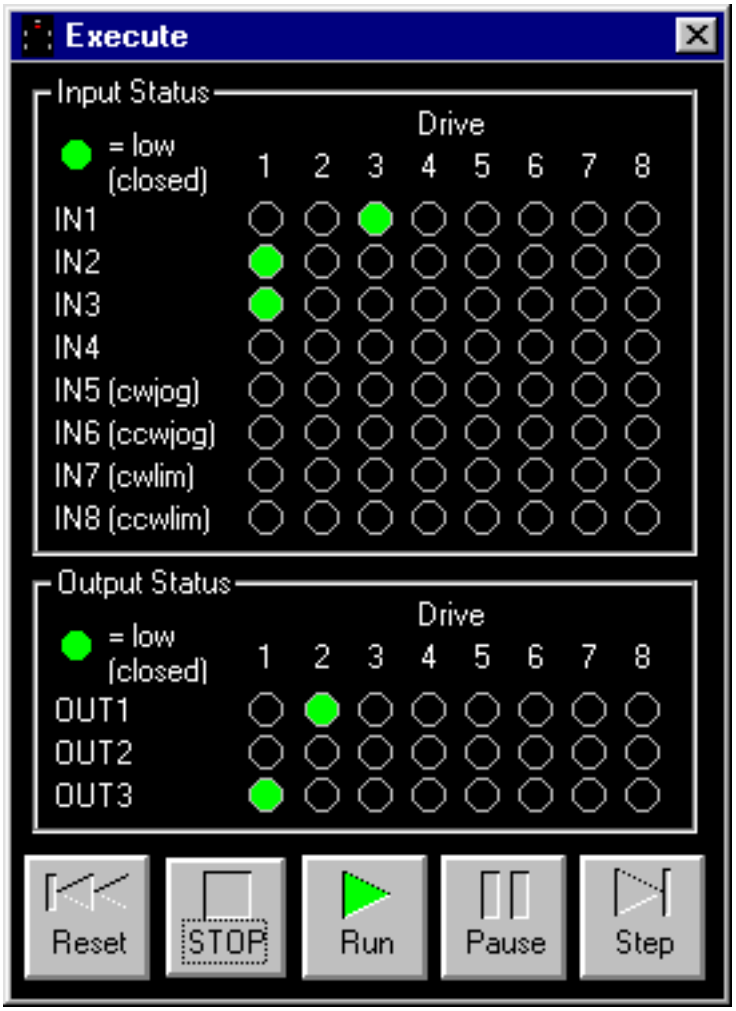
Pressing **STOP** will interrupt the hub at any point in the program and close the execute panel. You will find this feature useful if the drive starts doing things you didn't intend for it to do.

Pause halts the program, but does not close the execute box. While the program is paused, the display of inputs is continuously updated, so you can adjust sensors and switches and see the result in real time.

Step executes the next line of the program, then automatically pauses it again.

Pressing **Run** makes the program run again, from where you paused it.

Reset sends program execution back to line 1.



Save, Open, Print & Quit

In addition to exchanging programs with the SiNet™ hub, the programming software can also save & load programs using your hard drive, and can print hard copies of programs using your printer.

The Save button lets you save a program to the hard drive. A file dialog box will ask you to pick a name for the program. You can use the same types of file names (i.e. “long file names”) that you would use for other Windows 95 compatible applications.

The Open button provides you with a dialog box showing all the compatible files on your drive. Click to select one, then click OK to load it. You can open hub programs and programs designed for or saved by the single axis *Si Programmer™* software.

Several example programs are installed with your programming software. It’s a good idea to load some of the examples and look at them: they may help you with your own application.

Print lets you make a hard copy of your program on any printer that’s attached to your computer and installed in Windows. Print uses the standard Windows printer dialog, allowing you to specify which printer to use if you have more than one.

If you call for help, we may ask you to print your program and fax it to us. If you have a fax modem that acts as a printer, you can use it to fax the program directly to us from the programming software. You’ll have to enter our fax number at some point, which is (831) 761-6544.

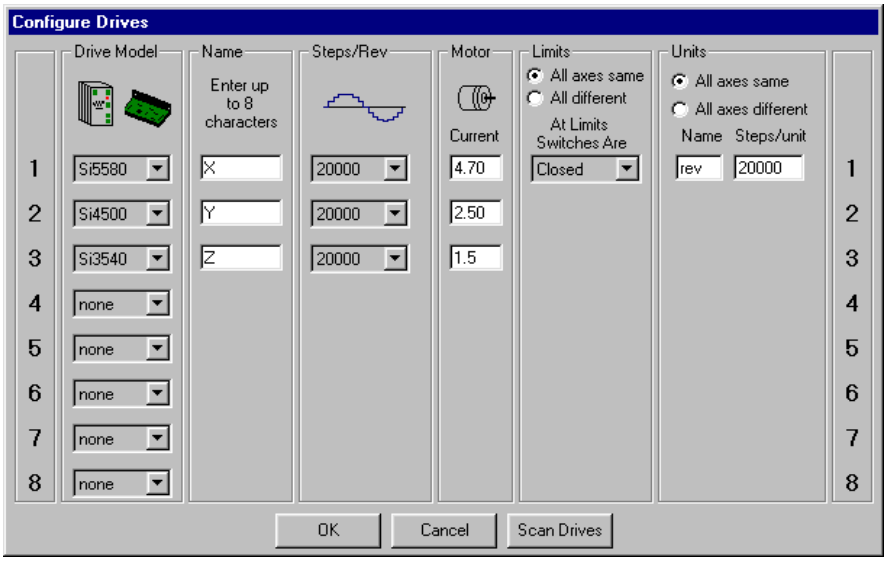
The Quit button exits the *SiNet Hub Programmer™* and returns you to the Windows desktop.

Defining Your Hardware

The best place to start preparing your SiNet program is by defining your hardware. The hub needs to know a few things, like which drives you'll be using and how much current the motors require.

That information is summarized on the left side of the screen, under the heading "Drives." If you click on the "Change..." button, you'll see the "Configure Drives" dialog.

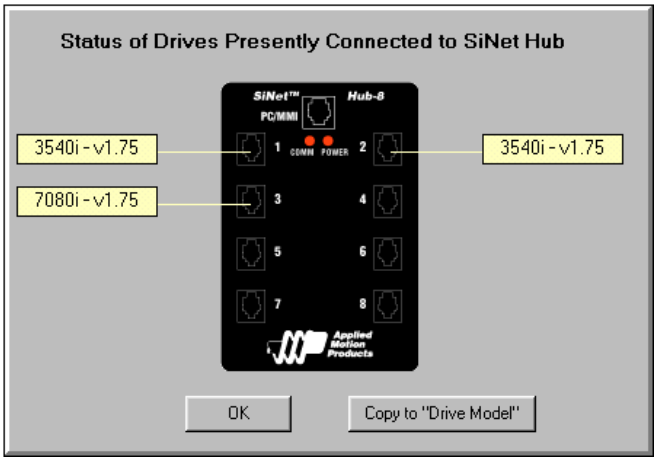
The first thing you should do is to select the drive models you plan to use for each axis. Select "none" for the axes you're not using.



If you already have the drives powered up and wired to the hub, there's a quicker way to get started. Just click on the "Scan Drives" button. You'll see a picture of the hub with descriptions of each of your drives.

If your drives are connected correctly and were powered up at the same time as your hub, then this information should be accurate. You can copy the data to the Configure Drives dialog by clicking on the "Copy to Drive Model" button. Then select OK to make the picture of the hub go away.

The SiNet Programmer software initially calls the axes "Axis 1", "Axis 2", etc. But you can type in names of your own, if you prefer. In the example above, we named our three axes "X", "Y" and "Z".



Next, choose the number of steps/rev for each drive. This may affect the way your unit definition works out. For example, if you have a linear motion system and want to work in inches, you'll need to state how many motor steps are in one inch. If you have 5 turn/inch screws, and set the steps/rev at 20,000, then you'll set the steps/inch as 100,000.

If you want to program the same system in millimeters, the steps/mm would be 5 * 20,000 / 25.4 = 3937.008. But that's not a round number, and would result in conversion errors. If you choose 25400 steps/rev, then you'd have 5 * 25400 / 25.4 = 5000 steps/mm. Much better. (Hint: if you have a rotary system and want to work in degrees, try 36,000 steps/rev.)

If you select the BL7080i servo drive, the list of predefined step resolutions will be replaced by a text box. For accurate move distances, you must enter the encoder counts/rev here. When you set up the motor with the *Quick Tuner* software, remember that value that was in the "Encoder counts/rev" box? That's the same number you need here.

Next, you'll need to enter the rated motor current for each axis. This setting may depend on how you have connected your motors to the drives (series or parallel, for example.) Please refer to the Hardware Manuals

that came with your drives for more information about motor connections and current settings.
Finally, we need to discuss limit sensors. Many linear motion systems employ limit sensors so that the load does not travel beyond it's intended limits and cause damage. You have three choices here: Closed, Open and Not Used. "Closed means that the limit switch or sensor is closed at the end of travel limit, and is normally open. "Not Used" indicates that you have no limit sensors and would like inputs 7 and 8 to be available for other purposes such as triggering, branching or other types of sensing.

Some users prefer sensors that open at the limit, because if a wire breaks or the sensors fails, you'll find out right away because a limit condition will be reported.

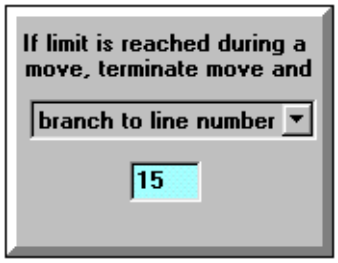
If you are not using limit sensors, choose "not used". That frees up the limit sensor inputs on the drive for other uses.

There are two options available for setting the limit sensors. If you choose "All axes same" (the default value) then one setting applies to all the drives in your system. If you choose "All different" then you can set each axis individually.

When you've got the drives fully configured, click on the OK button.

This is a good time to save your program to disk. If you click on the Save button, you'll see the standard Windows file dialog.

Reaction to a Limit



Since we finished the last section by talking about limits, this seems like a good time to discuss what happens when your program is running and one of the motors run into an end of travel limit.

You have three choices, which you choose from a list in the lower left corner of the main screen. No matter which option you choose, the motor that hit the limit is stopped immediately.

"Branch to a line number" gives you a chance take action in your program when something goes awry. You might, for example, want to branch to an MMI Prompt telling the operator to investigate the problem.

"Continue program" stops the motor that tried to go too far, but continues running your program as if nothing happened. This option is useful when you are first testing your program, but is not likely to be used "in the field."

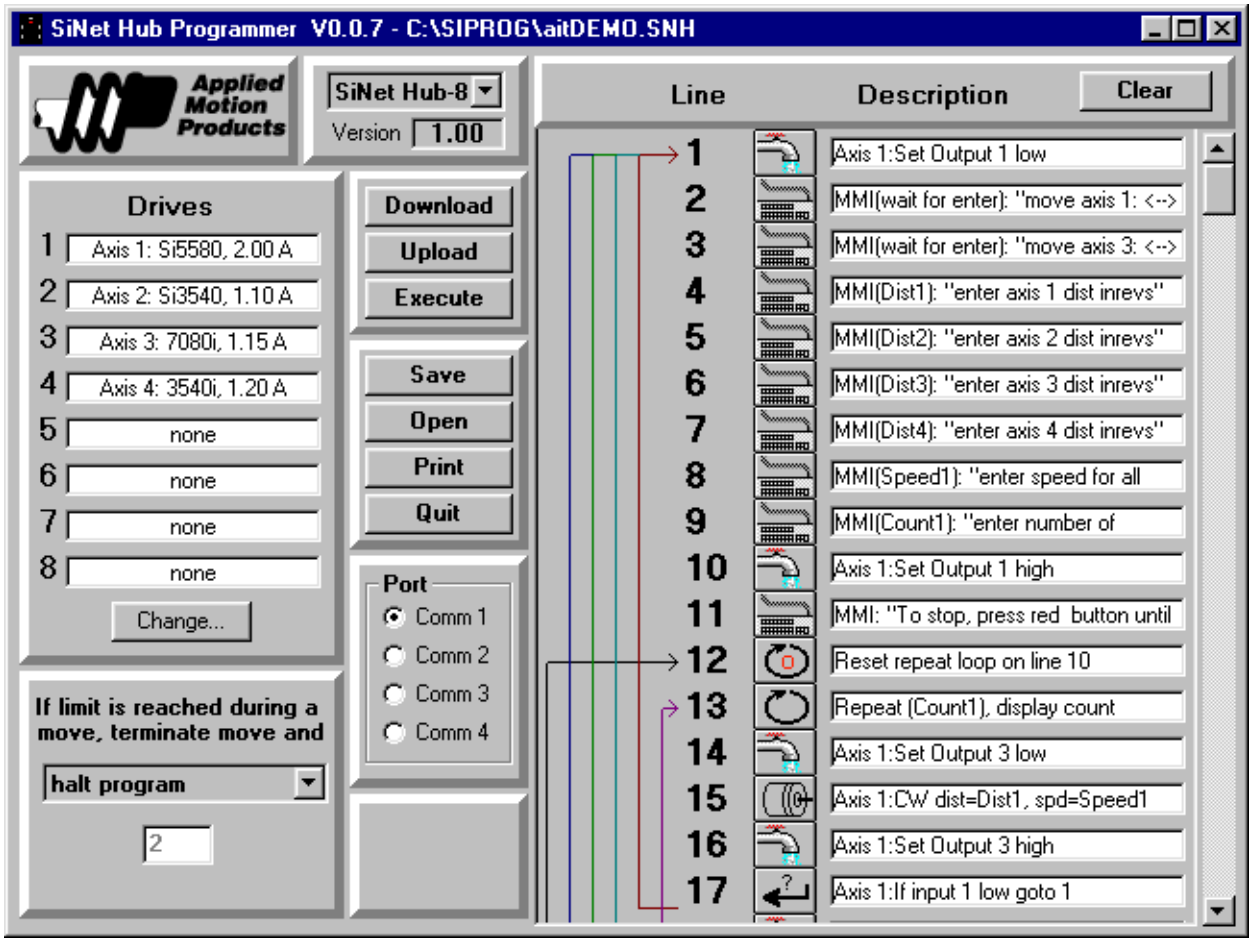
"Halt program" will freeze the execution of your program and flash the COMM light on the front of the hub. If the hub is connected to the PC for the purposes of writing and testing your program (we call this "development mode") then you will see a message telling you what happened. You'll also have the option of resetting the hub.

Entering Your Program

If you installed the software back in the section entitled “Installing the Programming Software” then you’re ready to go. If not, please go back and review that section.

It is also safe to assume that you’ve read the sections “Defining Your Hardware” and “Reacting to a Limit”. (You wouldn’t dare skip ahead to the good stuff, would you?)

We are now ready to put some instructions in our program and try it out.



To activate the software, click on the Start button, then Programs...Si Programmer. Choose SiNet Programmer. The main programming window will soon appear, as shown above. The title bar will display the *SiNet Hub Programmer™* software version.

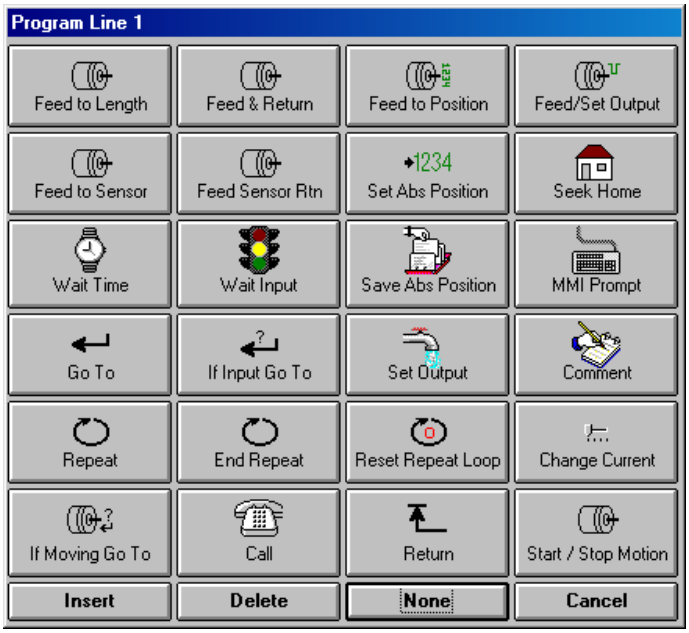
If you have a SiNet™ Hub connected to the PC, turn it on now. (The hub get it’s power from drive #1, so turn on that drive.) After you do, your computer should beep. The “Version” box will display the version number of the Si™ firmware that’s in your hub.

If you don’t have any Si™ hardware connected to your PC, you can still write programs.

Let’s try a simple program. Many programs will begin with the instruction *MMI Prompt.* That way when you first turn on the system, it doesn’t do anything until you tell it to. We’ll put a *Wait for MMI Enter* instruction on the first line.

Next to the large number 1 in the Program Window is a button showing the "nothing" icon. That indicates that there is no instruction for that line. Anytime the Si™ Indexer encounters a "nothing" program line, it simply moves on to the next line, as the icon implies with a downward pointing arrow. After the SiNet™ hub executes the instruction on line 200, it automatically jumps to line 1, unless the instruction on line 200 makes it jump somewhere else.

To enter an instruction on program line 1, click once on the program icon. You should now see the "Program Line..." dialog box, shown here.



Click on the button marked "MMI Prompt". The MMI Prompt dialog box will appear. Choose the option "Display text and wait for enter." Enter the phrase "Press enter to move" in the text box on the right. The click OK.

The first line of your program should now display the MMI Prompt icon, and the description "MMI (wait for enter)".

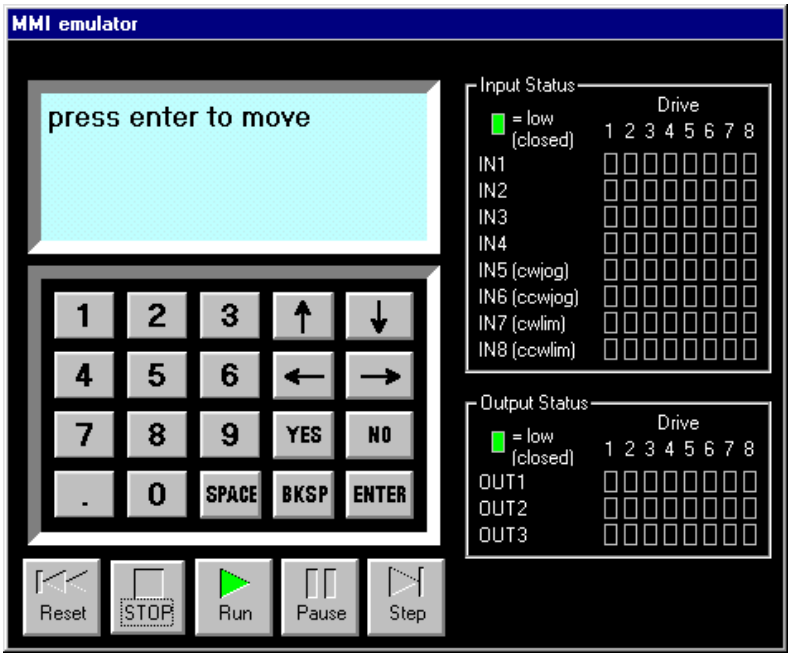
Click on the icon button for program line 2. This time when you see the "Program Line..." box, click on Feed to Length. In the Feed to Length box, enter the distance as 1 rev, then slide the speed bar to around 5 rev/sec. Click OK.

The second program line should now show the motor icon and the caption "Axis 1: CW 1 rev, 5 rps."

Next, click on the line 3 icon. Choose Go To. In the Go To dialog, the line number will probably already read "1". That's what we want. Click OK. We're ready to download and run.

I'm assuming that you already did your hardware set up. If not, please go back and review that section. We can't run your program unless the hub has accurate information about your drivers and the motors that are connected to them.

If your system is properly defined and powered up, press the Download button. Your program and your drive configuration will be transferred into the hub, where it will remain until you download another program. At this point, you may be wondering why we asked you to put an MMI Prompt in your program, when your PC is hogging the hub's MMI port. Perhaps you don't even have an MMI. Fear not: to simplify your programming process, we've included a "virtual MMI" in the SiNet software. Click on the Execute button, and you should see it.

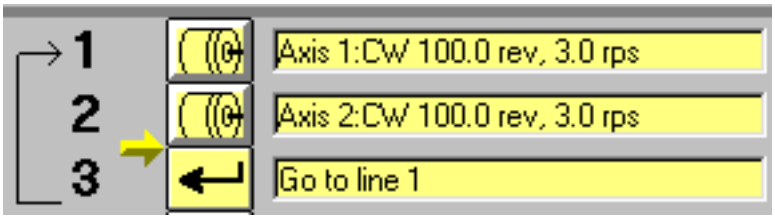


If you click on the fake MMI's ENTER key, the motor on axis 1 should make a short move.

That's about it for the basics. The next few pages will give you more techniques for editing your program. Then your job will be to learn all the instructions and figure out how to apply them to your application.

Animation

When your program executes while the hub is connecting to your PC, the Hub Programmer "animates" the execution on your screen. The program lines turn yellow as they execute, and the inputs and output are displayed in simulated LEDs. Since a hub-based system can do more than one thing at a time, you will sometimes see more than one "yellow line" at a time. Because this can be confusing when debugging your program (yes, you too will sometimes create a program that does not work as expected), we also display a "yellow arrow". The yellow arrow is the "program counter". That's the line that the hub is executing. Any other yellow lines are being handled by the individual drives. If your program is hung up at a particular spot, look at the instruction below the yellow arrow. That instruction needs something in order to proceed: an input to match a condition, an MMI entry, or a move to be completed.



Editing Tips

***Changing an Instruction**:* if you need to edit an instruction that's already in your program, and wish to go directly to the appropriate dialog box, hold down the shift key while you click on the instruction icon.

Copying Instructions: there may be occasions where you want to make an exact copy of an instruction, or perhaps a copy with only one or two parameter changes. The fastest way to copy an instruction from one line to another is to point the mouse at the instruction icon that you want to copy, and drag the icon onto another one elsewhere in the program. This is referred to as "Drag and Drop."

For example, say you've entered a *Set Output* instruction on line 5 of your program, and you'd like an identical *Set Output* on line 11. Position the mouse over the *Set Output* icon on line 5, then click and hold the mouse button. Move the mouse until the icon is over the line 11 icon. Let go of the button, and the instruction with all of its parameters will be copied to line 11.

Inserting New Program Lines: the time will no doubt come when you've entered many program lines only to realize that you need to add an instruction right in the middle. We could be cruel and tell you that you'll have to reenter most of the instructions to make room for the new one. But, in the spirit of making the Si™ Indexer easy to use, we've included a command to insert a new instruction anywhere in your program.

It's easy to do: just click on the program icon where you want the new instruction to go. You'll get the Program Line... dialog, as usual. Instead of choosing one of the 20 instructions, click on the command button marked "Insert." The instructions are reordered, and the line you need for your new program line is available. You can then click on the new line and select an instruction as you normally would.

Deleting Program Lines: in addition to inserting a line in your program, you can delete one to make room for others farther down. You may, for example, have some available space in the middle of the program, but want to add an instruction near the end.

Click on a program line that you don't need. When the Program Line... dialog comes up, select "Delete." The line you selected for deletion will be gone, and all the other lines will move up one position, leaving a blank spot at the end of the program.

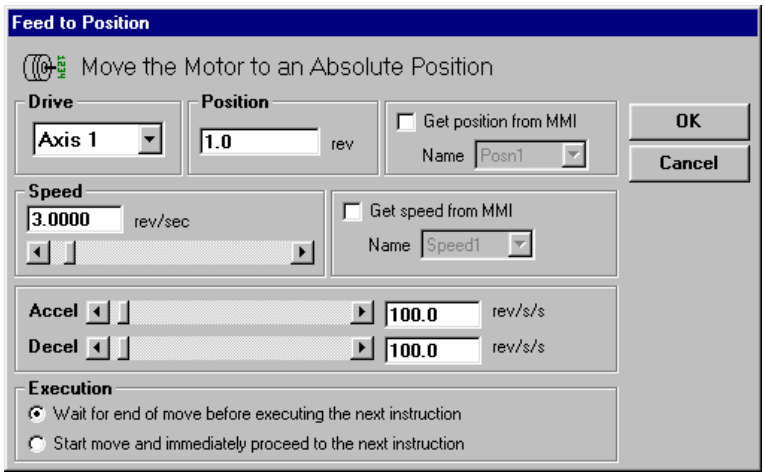
By combining insertions and deletions, you can place your program lines wherever and whenever you need to.

The Instructions

The next pages describe each of the 22 instructions that you can use to build your SiNet program. Before you proceed, we need to discuss the order in which the instructions execute.

If you've used the single axis Si products before, you're probably used to the program running instructions one at a time, with no instruction executing before the previous one has finished.
Because a SiNet system has a processor in the hub plus processors in each drive, it's natural for more than one thing to happen at once. At the very least, you might want to move two axes at the same time. Or maybe, when your machine is first powered up, you want all the drives to home themselves while the operator answers a series of MMI Prompts. Both of those things, and more, are possible. You just need to be aware of which instructions can execute at the same time as others.

The motion instructions (Seek Home, Feed to Length, Feed & Return, Feed to Position, Feed to Sensor and Feed to Sensor & Return) each give you three options. If you choose *Wait for End of This Move before Executing the Next Instruction*, the hub will not execute the next line of your program until the move is complete. If you choose *Wait for End of All Moves before Executing the Next Instruction*, the Hub will not execute the next program line until all motors are stopped.



If you choose *Start move and immediately proceed to the next instruction*, then the move and the instruction on the next line can take place simultaneously. If the next line is another move (on a different axis, of course), and it is also designated as "start and proceed", then a third line of your program can execute with the two moves. In fact, you can run all axes at the same time. (All simultaneous moves start within 1/2 millisecond of each other.)

You may be wondering what happens to your program if all the moves are the "start and proceed" type. That depends on the instructions that follow the moves. Some instructions "pass through" automatically. Wait Time, Set Output and MMI Prompt as pass through instructions. They execute regardless of any ongoing motion. When they are done, the next line in your program executes.

Some of the other instructions can pass through if you want them to. For example, if you follow two concurrent moves with a Go To, the Go To will not take place until both moves are finished if you check the box "wait for all moves to finish". If you do not check this box, the hub will execute the GoTo instruction during the moves.

These features give you the opportunity to:

- Run one or more motors while setting or pulsing an output
- Run the motors while interacting with the user via the MMI
- Start one move, wait a certain amount of time, then start a move on another axis
- Cycle one motor back and forth while another runs continuously
- Operate two drives "independently" while sharing an MMI.

Now that we've got that straight, let's learn about the instructions.

MMI Prompt

The Si™ Indexer is available with an optional MMI (Man Machine Interface), sometimes called an operator panel. The MMI attaches to the same RS232 port that you use to connect to your PC, using the same cable. The MMI has a four line liquid crystal display (LCD) and 20 keys for entering data. There are seven things you can do with the MMI:

1) *You can display a message on the LCD*. You might want to identify your machine ("ABC Bottle Filling Co. Model 20") or display a status message ("Machine Running - Status OK").

2) *You can pause your program until the user presses ENTER*. For example, if you were applying preprinted labels, eventually you'll want to halt the process until the operator loads a new roll of labels.

3) *The MMI can ask the user to make a decision*. For example, you might want to offer the user an option, like changing set up parameters, that can be responded to by pressing the yes or no keys.

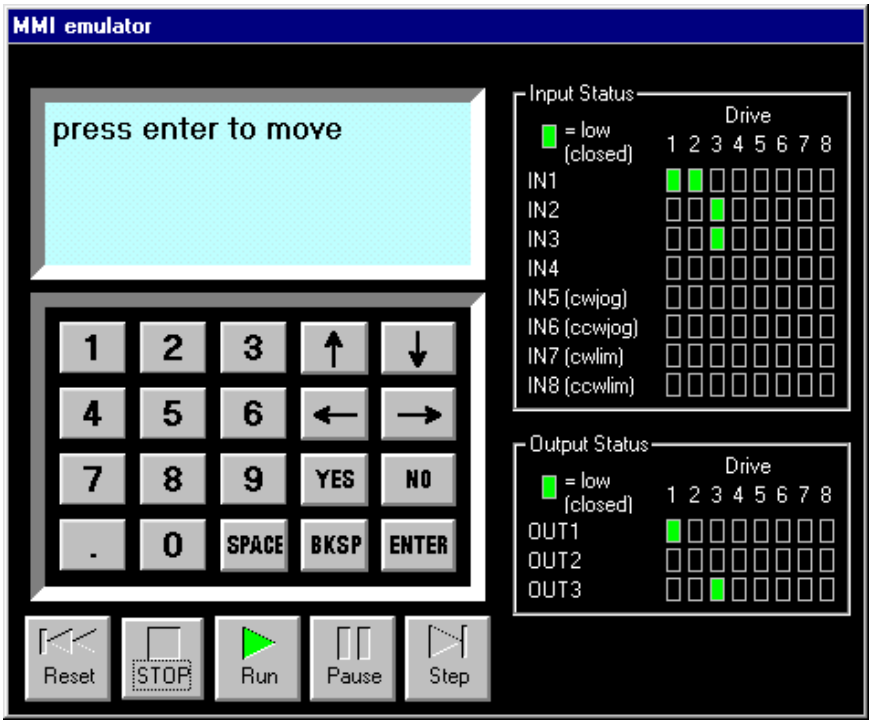
4) *The user can be asked to enter a move distance*. If you want to build a machine that feeds out material and then cuts it off, the operator can specify how long the resulting material will be.

5) *The user can be asked for a move speed*. This option allows the operator to adjust a feed rate, flow rate or other motor speed related setting.

6) *The user can be asked for a repeat count*. You can let the user set the number of parts that are processed. You can also combine a repeat loop with a *Wait Time* instruction to adjust dwell time.

7) *You can display a menu, wait for the user to press a numeral key, then branch to a corresponding program line.* Any or all of the keys 1 - 8 can be used, each with it's own branch address.

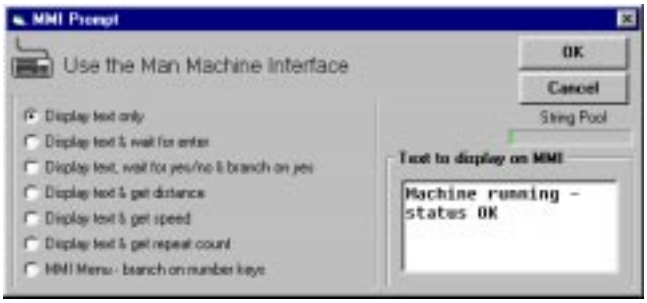
If you are thinking ahead at this point, you might ask "If the MMI plugs into the same port as the PC, how can I run a program from the PC that uses the MMI?" Like most features of the *Si™* software, it's simple. If you press the Execute button and the program in your drive contains any MMI instructions, you will see a different control box on your screen. The MMI control box looks and acts like the real MMI: it will display messages, and you can click on the buttons to enter data. Like the other execute box, there is a display showing the status of inputs and outputs, and a control panel that allows you to interrupt, single step, or restart your program at any time.



We provided the emulated MMI to save you the expense of a second RS232 port on your SiNet™ Hub. It also allows you to try out the MMI before buying one.

How to display a message on the MMI

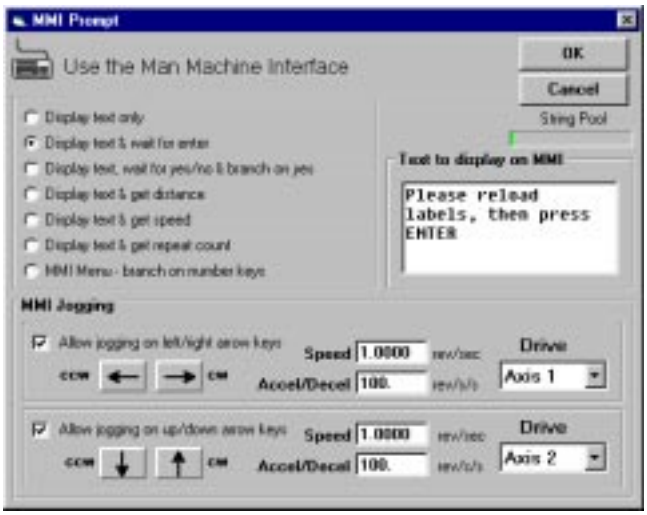
1) Type the message that you want to display in the text box. Example: "Machine Running Status OK" in the text box
2) Select the "Display Text Only" option button
3) Click the OK button
Note: The message will stay on the LCD until another instruction uses the MMI. This is useful when you want set a Repeat instruction to display the loop count on the MMI.


MMI Prompt
Use the Man Machine Interface
OK
Cancel
String Pool
Display text only
Display text & wait for enter
Display text, wait for yes/no & branch on yes
Display text & get distance
Display text & get speed
Display text & get repeat count
MMI Menu - branch on number keys
Text to display on MMI
Machine running -
status OK

How to pause until the user presses ENTER

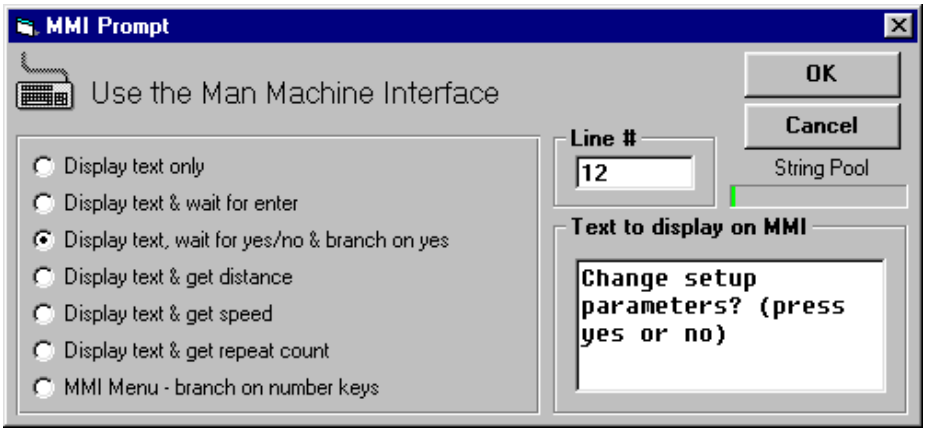
1) Enter your message in the text box. Example: "Please reload labels, then press ENTER"
2) Select the "Display text & wait for enter" option button
3) If you want the user to be able to jog a motor using the MMI left and right arrow keys, check the box marked "Allow jogging on left/right arrow keys" , then select a drive from the list. You'll also need to set the jog speed and jog accel/decel rate.
4) If you want the user to be able to jog a motor using the MMI up and down arrow keys, check the box marked "Allow jogging on up/down arrow keys" , then select a drive from the list. Set the jog speed and jog accel/decel rate for this axis.
5) Click OK

Note: If you want to let the operator position a load precisely, but also traverse large distances, you can do it with two MMI Prompts. Set the jog speed fairly high on the first one (rapid traverse), and low on the second (fine tuning).



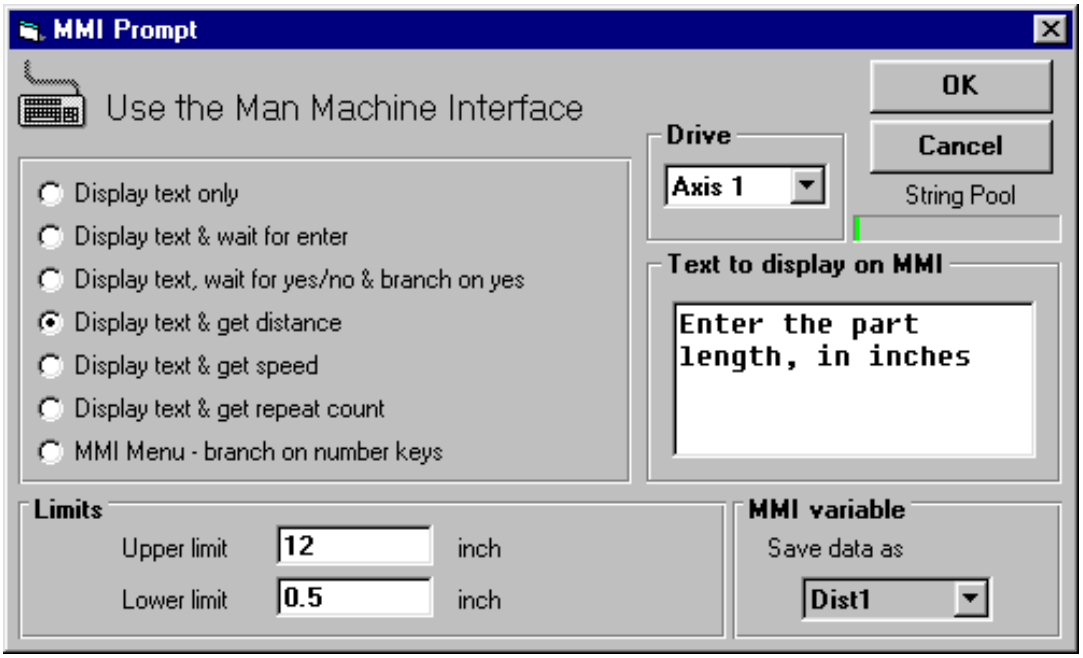
How to let the user make a decision (MMI branching)

1) Type your message into the text box. Example: “Change setup parameters? (press yes or no)”
2) Select the option button “Display text, wait for yes/no & branch on yes”
3) Which program line to you want to branch to if the operator presses YES? Put that number In the line # box
4) Click OK


MMI Prompt
Use the Man Machine Interface
OK
Cancel
String Pool
Line #
12
Text to display on MMI
Change setup parameters? (press yes or no)
Display text only
Display text & wait for enter
Display text, wait for yes/no & branch on yes
Display text & get distance
Display text & get speed
Display text & get repeat count
MMI Menu - branch on number keys

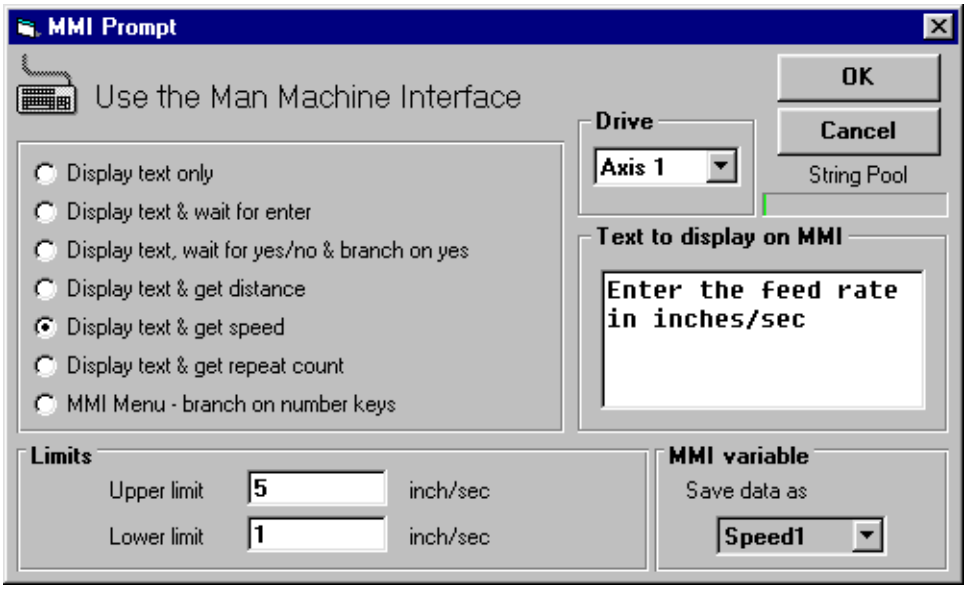
How to ask the user for a move distance

1) Type “Enter part length, in inches” in the text box
2) Select the option button “Display text and get distance”
3) Which axis will you use this distance for? Select it from the list.
4) Enter upper and lower limits (in this example, we only want to allow the operator to enter distances between 0.5 and 12 inches)
5) Select an MMI variable to store the distance in (we chose Dist1 this time, but any of the 50 MMI variables is acceptable for storing any type of data)
6) Later in your program, you’ll need a motion instruction (Feed to Length, Feed & Return, etc) that uses the Dist1 variable for distance.



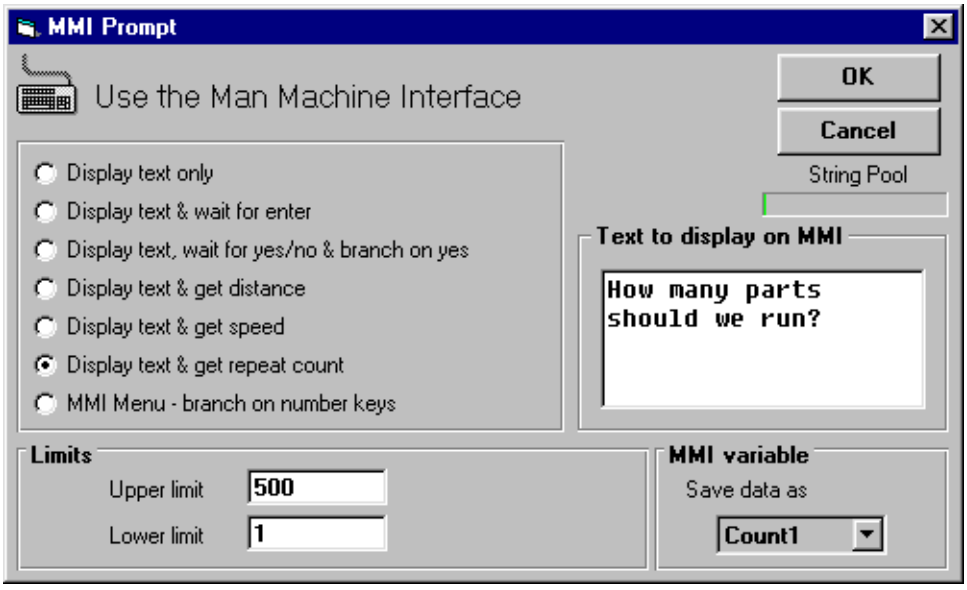
How to get a speed from the user

1) Type “Enter the feed rate, in inches/sec” in the text box
2) Select the option button “Display text and get speed”
3) Enter upper and lower limits (in this example, we want to allow the operator to enter flow rates between 1 and 5 inch/sec)
4) Select an MMI variable to store the speed in (we chose Speed1)
5) Later in your program, you’ll need a Feed instruction (Feed to Length, Feed & Return, etc) that uses the Speed1 variable for speed.



How to get a repeat count from the user

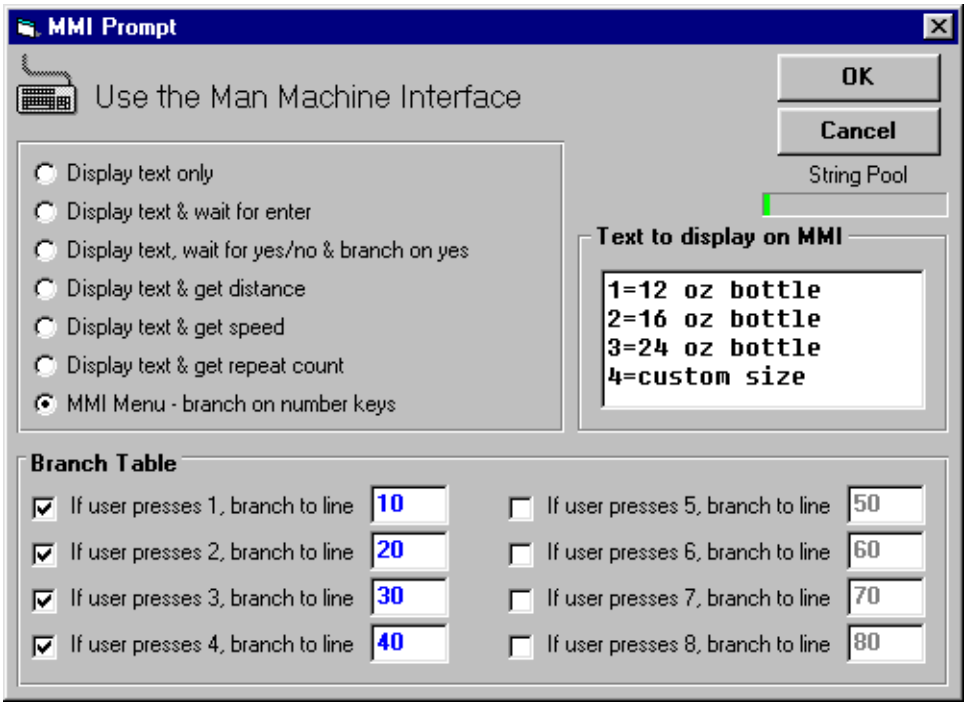
1) Type “How many parts should we run?” in the text box
2) Select the option button “Display text and get repeat count”
3) Enter upper and lower limits (in this example, we want to allow the operator to enter a number between 1 and 500)
4) Select an MMI variable to store the count in (we chose Count1)
5) Later in your program, you’ll need a Repeat instruction that uses the Count1 variable for the repeat count.



How to create an MMI Menu

- Select an MMI Prompt instruction
- Select the option button “MMI Menu...”
- Type your menu text in the text box (you can enter up to four lines)
- Check the boxes indicating which numeral keys you want to use (in this example, we used 1,2,3 and 4)
- Assign a program line number to each key (we used 10, 20, 30 and 40)
- Later in your program, you’ll need to put instructions at each of the lines you’ve specified. These are the instructions that will execute if the user presses the appropriate key on the MMI. For example, when the user presses ‘1’, the program will branch to line 10.

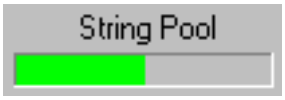
Note: if you add extra spaces to your display text to get the look “just right”, watch out: sometimes the *SiNet Hub Programmer™* removes those spaces when you reopen the instruction dialog. It is safer to format your text using other characters, like ‘.’ (period) or ‘_’ (underscore).


MMI Prompt
Use the Man Machine Interface
OK
Cancel
String Pool
Display text only
Display text & wait for enter
Display text, wait for yes/no & branch on yes
Display text & get distance
Display text & get speed
Display text & get repeat count
MMI Menu - branch on number keys
Text to display on MMI
1=12 oz bottle
2=16 oz bottle
3=24 oz bottle
4=custom size
Branch Table
If user presses 1, branch to line 10
If user presses 2, branch to line 20
If user presses 3, branch to line 30
If user presses 4, branch to line 40
If user presses 5, branch to line 50
If user presses 6, branch to line 60
If user presses 7, branch to line 70
If user presses 8, branch to line 80

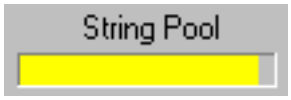
What is the String Pool?

In addition to storing your 200 program lines and up to 50 MMI variables, the SiNet hub must store the messages that you want it to display on the MMI. That area of memory is called the “string pool”, and it can hold up to 1500 characters.

If you use a lot of MMI Prompts in your program, and if the messages you want to display are long, you could actually run out of space in the string pool. To let you know how much space you’ve used, the MMI Prompt dialog includes a “string pool meter.” When the meter is green, you’re in good shape. It turns yellow when the pool is 90% full. If you see the meter turn red, you have less than 80 characters of space left.



About 50% Full



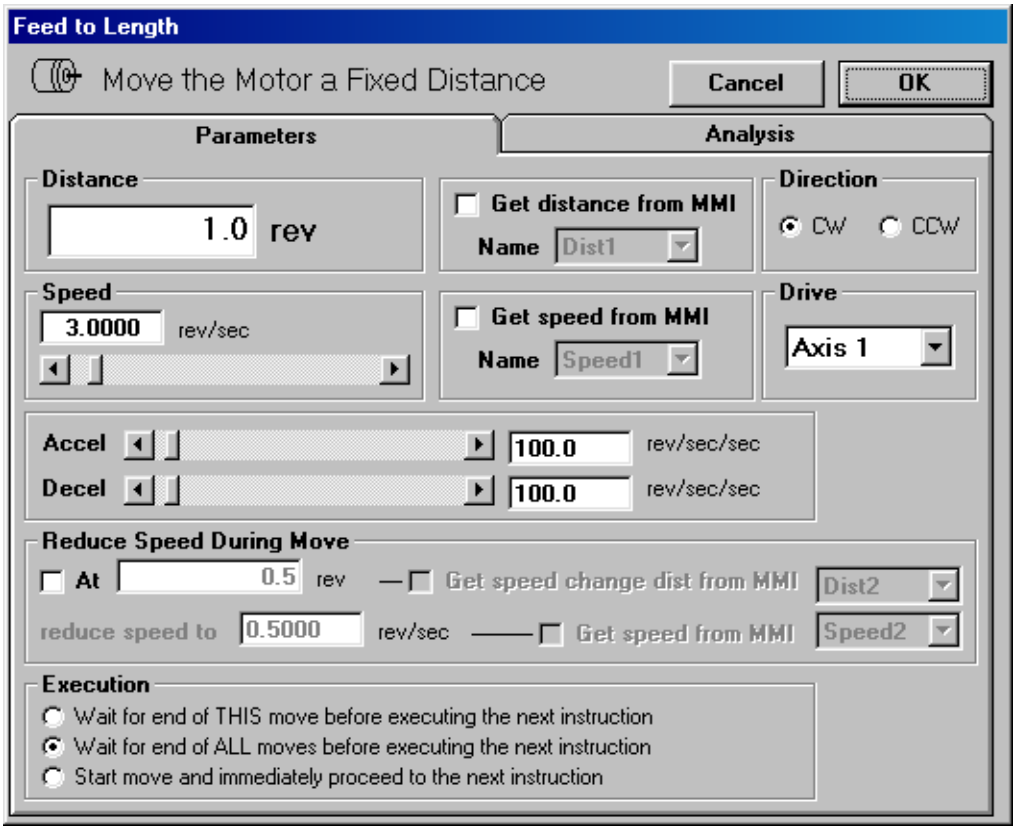
80% Full

Less than 80 chars left

Feed to Length

The *Feed to Length* instruction is used for point to point moves. If you just want to move the motor a fixed number of steps, this is the instruction to use. You can also use speed or distance data that was previously gathered by an *MMI Prompt* instruction.

When you click on the *Feed to Length* button in the Program Line... dialog box, you'll see the *Feed to Length* dialog box appear. This is where you enter the parameters for the move.



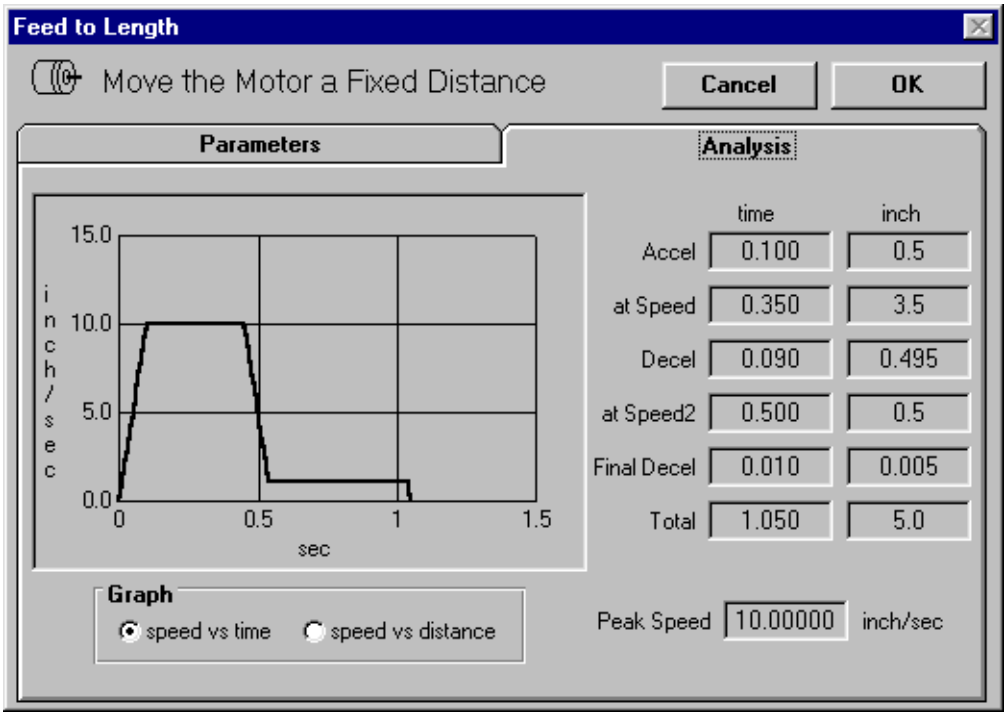
Distance - this is the distance you want to move. The maximum distance is 16,000,000 steps, but you'll enter the distance in the units you defined when you configured your drives. If you select the check box marked "Get distance from MMI", you can choose one of the 50 MMI variables as the distance. Please note that checking "Get distance from MMI" does not automatically make the Si™ Indexer stop and ask the user for an entry. You'll need an *MMI Prompt* instruction somewhere else in your program for that.

Speed - this is the maximum speed you want the motor to go. You can set the speed anywhere between .025 and 50 rev/sec, in increments of .025 rev/sec. The speeds you see will be in the units you defined in the Configure Drives dialog. If you select the check box marked "Get speed from MMI", you can choose one of the 50 MMI variables as the speed.

You can also reduce speed during the move by checking the box marked "Reduce speed during move." This is useful for applications where a tool may need to approach a work piece quickly, but slow down just before making contact.

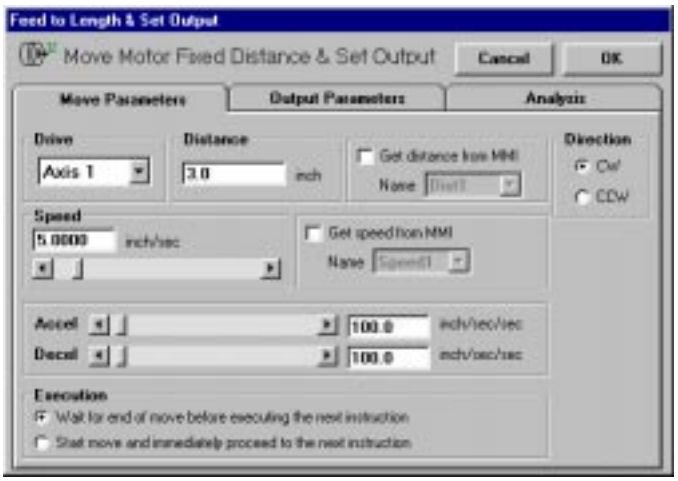
Direction - you can choose cw or ccw as the direction for the move. Just dot the appropriate circle by clicking on it.

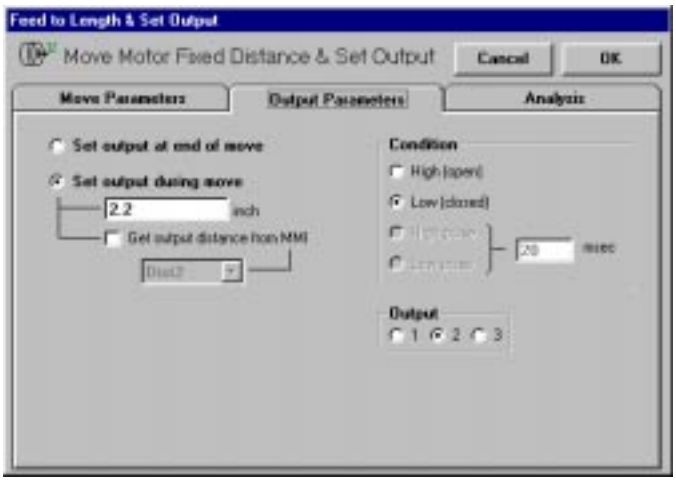
Analysis - Click on this tab to see a speed vs time graph of your move. It also provides some useful statistics about the move, such as the duration of the move and the portion of time spent accelerating and decelerating. You can also select a plot of speed vs distance.



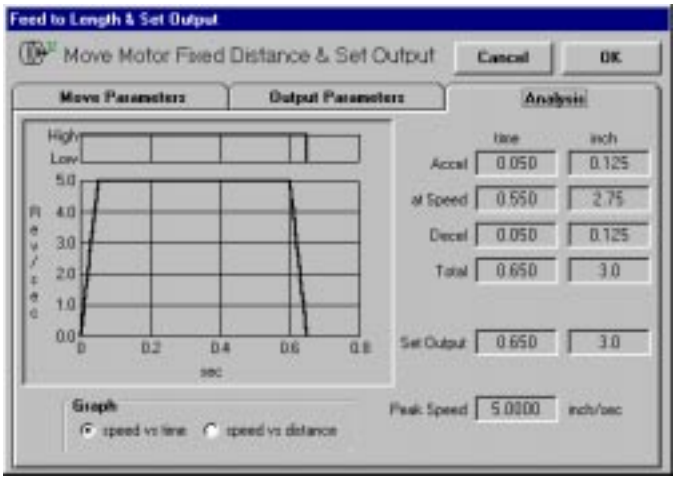
Feed & Set Output

The Feed and Set Output instruction is provided for two reasons. First, it allows you to combine a Feed to Length instruction with a Set Output instruction, making your program shorter. Feed to Length and Set Output are frequently used in combination because you want your Si™ product to signal another device when it finishes a move.





The second reason the *SiNet Hub Programmer™* has a Feed & Set Output feature is for manufacturing throughput. The Si™ indexer may be advancing a part which will then be processed by another device, for example feeding material to be cut off by blade. If the blade requires a little time to approach the material, you'd like to trigger the blade before you're done advancing the material. That way, you can process more pieces in the same amount of time.



Feed & Set Output allows you to set an output high or low at any point during the move. **The output must be on the same drive as the motor.**

Feed & Return

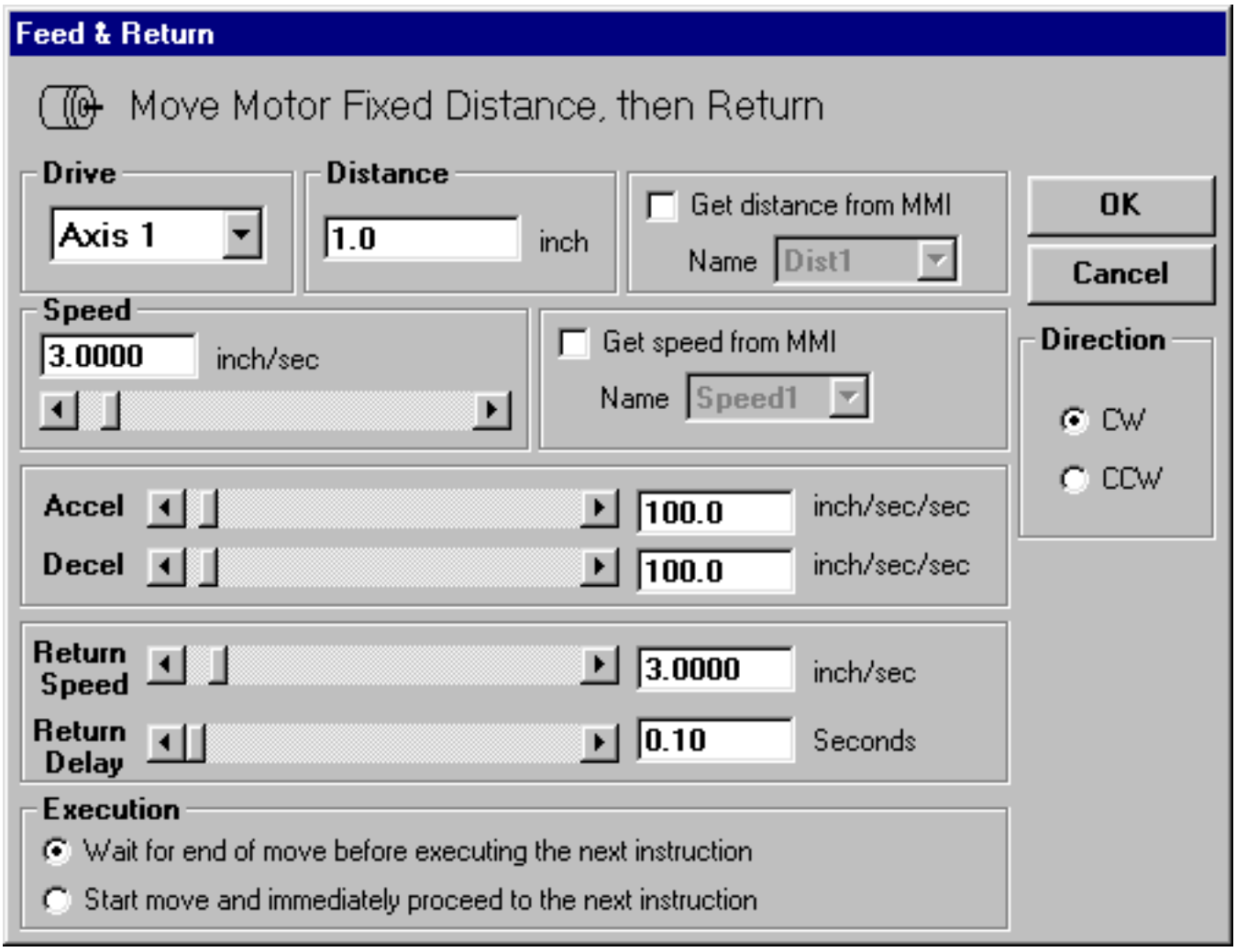
The *Feed & Return* instruction is used for point to point moves where you want to return to the starting point.

For example, if the motor was driving a cutoff knife, you would want to retract the knife after cutting.

Feed & Return requires many of the same parameters as *Feed to Length*: distance, speed, accel, decel and direction. For explanations of these, please refer to the *Feed to Length* section of the manual.

You'll also need to set the **return speed**. In the case of the cutoff knife, you might want to feed slowly, as the knife is cutting, then retract quickly. Thus, you would set the return speed higher than the forward speed.

Return delay determines how long the Si™ Indexer waits between the end of the feed move and the start of the return. This could, for example, give the machine time to remove a part before retracting. Since a motor and load need time to "settle out" after moving, you should not set return speed to less than 0.2 seconds unless you are certain that your motor and load settle more quickly than normal.



Feed to Sensor

The *Feed to Sensor* instruction allows you to move the motor until an external event changes the state of an input.

One useful application for *Feed to Sensor* is when your motion distance varies. Let's say you are using a step motor to dispense labels that come on a roll. You can't guarantee that the spacing of the labels is exact, so you don't want to simply feed out the same number of steps each time. Instead, you can put a sensor on the feed mechanism that "sees" the edge of each label and signals one of the Si™ Indexer inputs to stop motion.

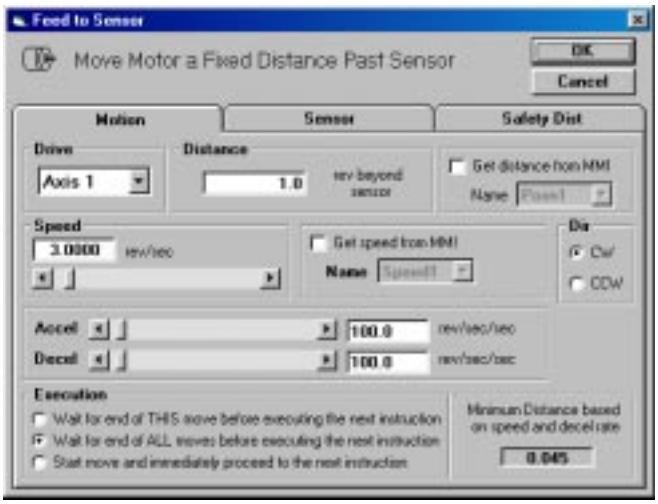
Feed to Sensor will ask for many of the same parameters as the other feed programs: Speed, accel, decel and direction. You also need to specify a distance. That's because the Si™ Indexer must have enough space to decelerate to a stop once the sensor is tripped. The higher the speed, the longer it will take to stop. If the decel rate is increased, then the motor can stop in fewer steps. The "Minimum Distance" box tells you how many steps you must allow, based on the speed and decel rate that you've set. You can't set the distance to less than this minimum.

You'll also need to tell the Si™ Indexer which input the sensor is wired to and what input condition to look for. **The input must be on the same drive as the motor.** The four input conditions are:

High - move until the specified input reaches a *high* signal state. This is the default state of an input if nothing is connected to it.

Low - move until specified input is at a *low* signal state.

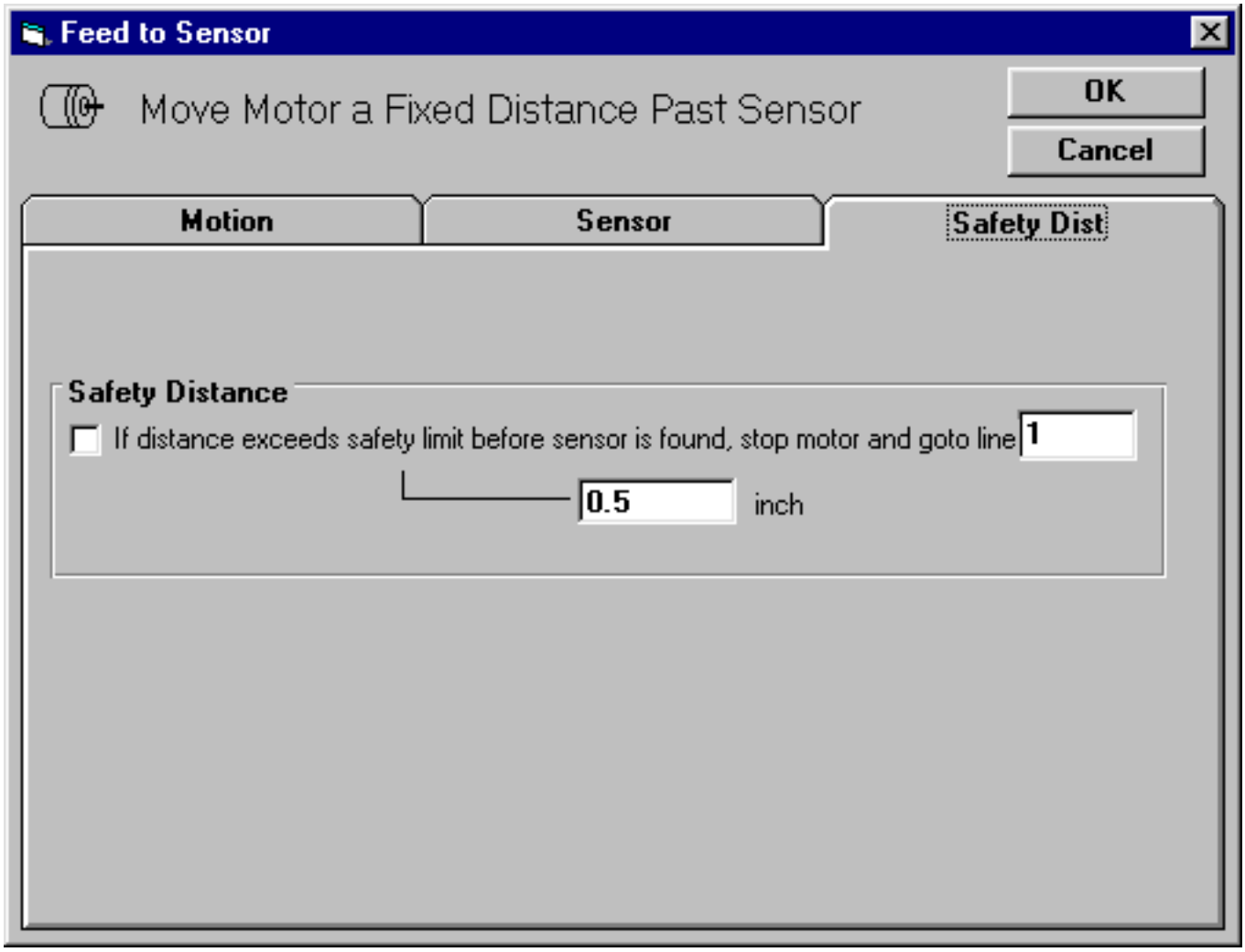
Rising Edge - move until the signal goes from low to high. This is similar to the high condition, but the difference is important. Let's say that you have a sensor wired to the Si™ Indexer that will go high when you want motion to stop. However, the sensor signal stays high after motion is complete, going low sometime later. This often happens in labeling applications where there isn't much space on the roll between labels. If you choose high as your input condition, the Si™ Indexer will complete the motion, then refuse to start again because the input signal is still high. If you choose rising edge, the Si™ Indexer would proceed with the input voltage high and stop when the sensor signal goes from low to high again.



Falling Edge - the opposite of rising edge. Si™ Indexer waits for an input voltage to go high, then low.

If you are concerned about your load never reaching the sensor (for example, if your sensor may fail or the load might jam up), check the box marked “If distance exceeds safety limit...”. You can then enter a safe distance and specify a line number to which the program will branch if it can’t find the sensor. For example, if you are sensing labels on a roll, and they are supposed to be about 1 inch apart, enter a safety distance of 3 inches. Then enter 20 as the branch line. On line 20, you will put some kind of error recovery routine, like an MMI Prompt telling the operator to check the label stock.

Note: the SiNet™ Hub Programmer software will not let you enter a move distance that is less than the minimum deceleration distance for your chosen speed and decel rate. However, if speed or distance is set by an MMI variable, no such error checking is performed. It is your responsibility to choose an upper limit of speed and lower limit of distance so that an operator cannot enter an incorrect value. Failure to do this may result in unexpectedly long moves.



Feed to Sensor & Return

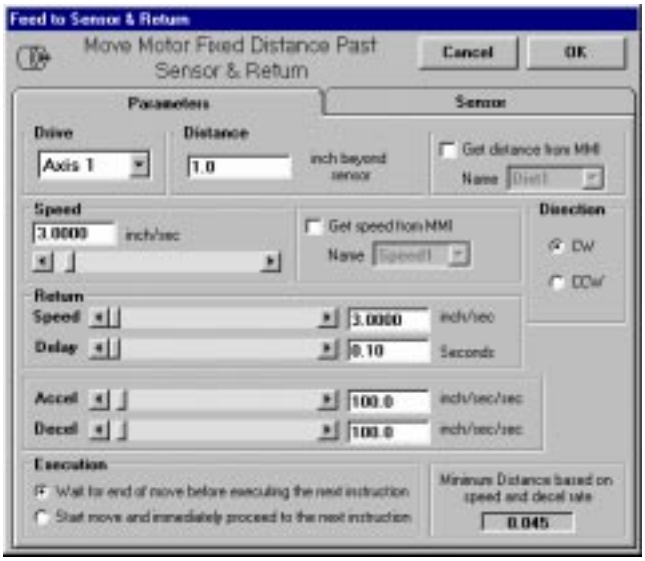
The *Feed to Sensor & Return* instruction is just like *Feed to Sensor*, but after the move the motor returns to the starting point.

Most of the parameters are the same as *Feed to Sensor*, but two new ones are added: the return speed and the return delay.

One useful application of *Feed to Sensor & Return* is a variable distance application. If we were building a machine to cut fabric of different sizes, and the Si™ Indexer was driving the cutoff knife, we might want to set a sensor at the end of the cut off stroke. That way, we can manually adjust for the width of material we happen to be using on a particular day without having to reprogram the Si™ Indexer.

Each time it's triggered, the indexer-drive would feed until the knife trips the sensor, then return to the starting point.

You should beware of one thing: the maximum distance for any move is 16,000,000 steps. That's the longest distance an Si™ Indexer can track. If you move more than 16 million steps before you hit the sensor, the Si™ Indexer will not return to the correct position. If this is a problem for you, consider selecting a lower microstep resolution. At 50,000 steps/rev, you would exceed the 16 million step limit after 320 revolutions. At 2000 steps/rev, you can go 8000 revs before exceeding the limit.

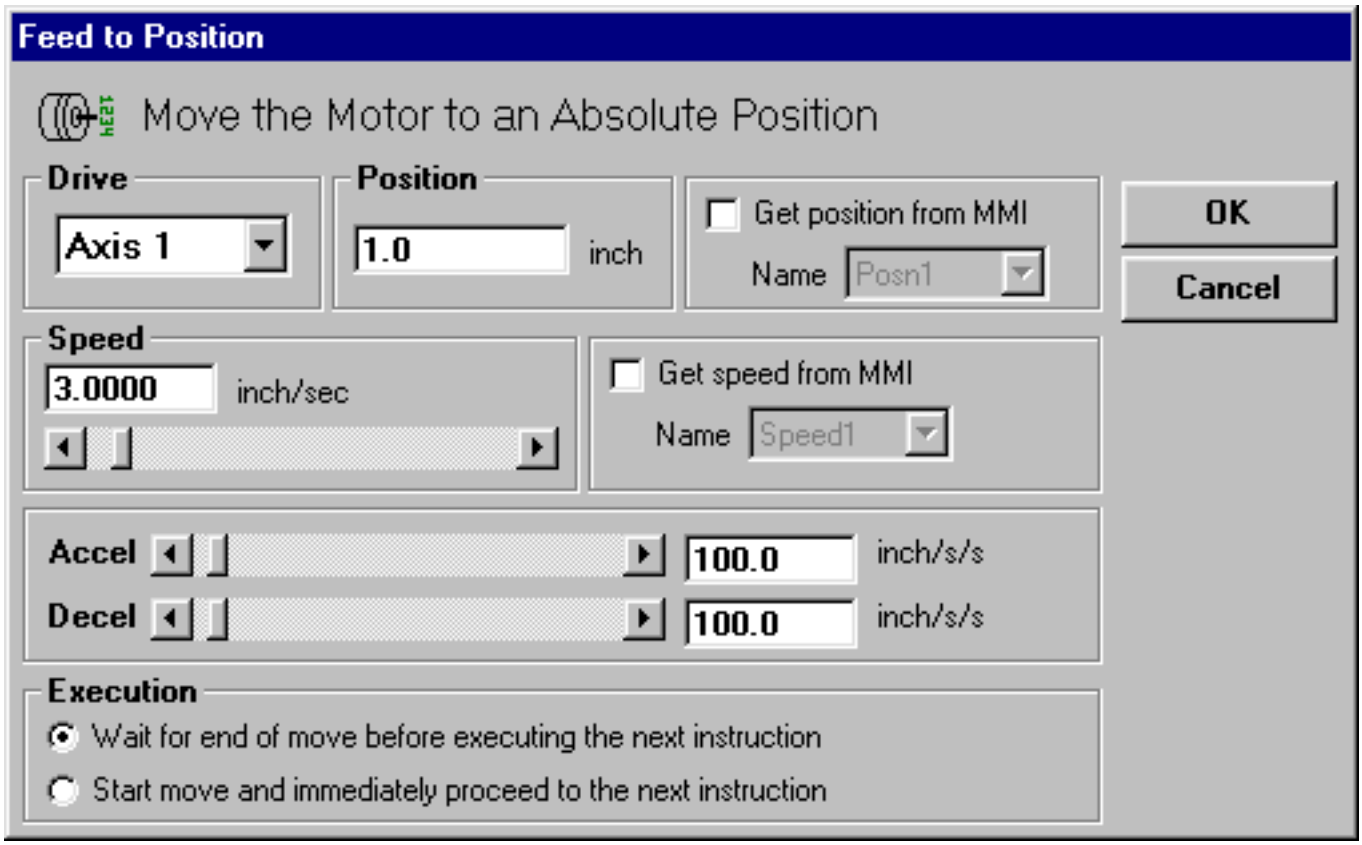


Feed to Position

This instruction moves the motor and load from wherever they are to an absolute position. For example, if the load is at the 4 inch position and the program executes a Feed to Position 6 inches, the motor will move two inches clockwise. If the load was at the 10 inch position and you did a Feed to Position 6 inches, the move would be 4 inches counter clockwise.

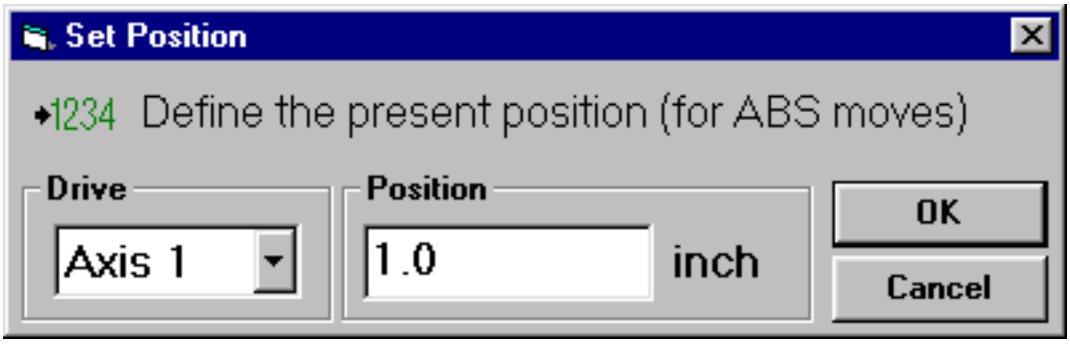
Feed to Position requires the usual move parameters: speed, accel & decel rates. Like other move instructions, speed can be recalled from an MMI variable, allowing it to be entered by the operator on the MMI panel.

Position can be a positive or negative number, and can be entered on the MMI. Please note that the MMI has no minus (-) key, so you can't enter a negative number on the MMI. You can avoid using negative absolute positions by using the Set Position instruction.



Set Abs Position

This instruction allows you to define the present motor position as any absolute position you like. The Seek Home instruction automatically clears the absolute position counter when it's finished, defining the home position as 0. But you may want something else. Perhaps you want to think of the home sensor as being the 8 inch position, or 90 degrees, or whatever. Simply put a Set Position instruction after the Seek Home instruction, or anywhere else in your program where you want to define the absolute position.



Save Abs Position

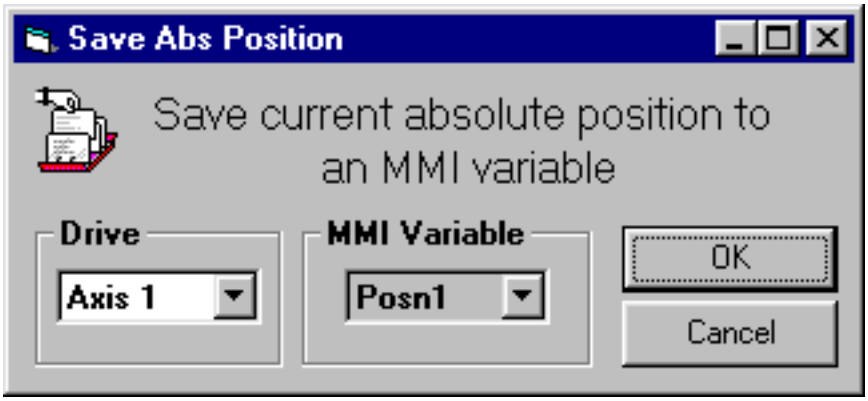
This instruction allows save the present absolute position to an MMI variable. This is useful is you want an operator to visually position your load, and then be able to return to that position later in your program.

For a material handling application, you could create a program that uses the Wait Input or Hand Wheel commands to allow the operator to move your load into position. The operator would then press an ENTER button to exit the Wait Input or Hand Wheel instruction. If the next instruction is Save Abs Position, the load position that the operator carefully obtained is recorded in nonvolatile memory. Elsewhere in your program you can use a Feed to Position instruction to return the load there.

Since the SiNet™ Hub supports up to 50 MMI variables, you can save as many as 50 different positions.

The sample program “LPdemo” demonstrates the “learning” of two positions.

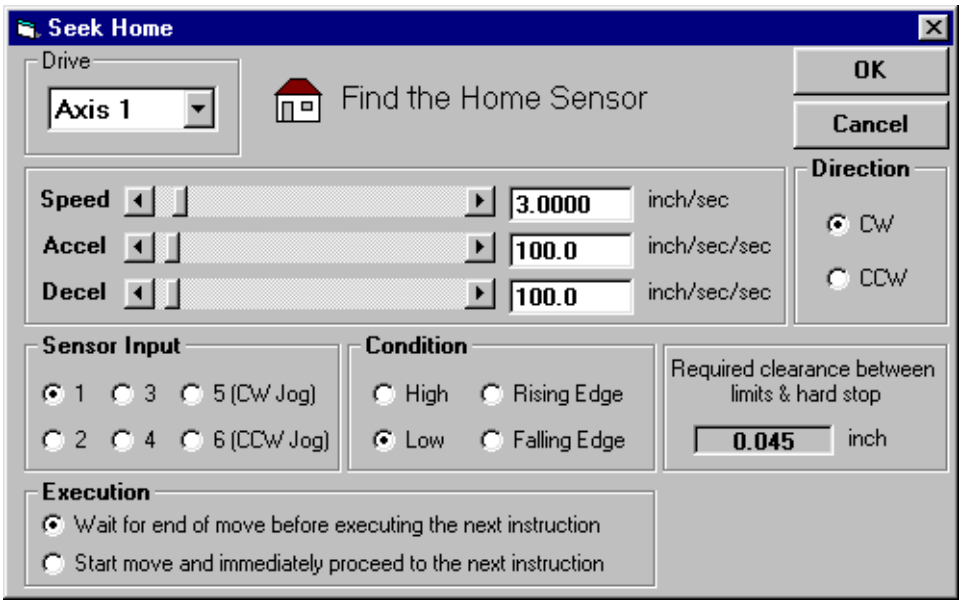
Even though the positions that the indexer has “learned” will still be remembered the next day (because they are stored in nonvolatile memory), you will need to “home” the system each time it’s powered up. Otherwise, the absolute positions that you’ve saved don’t make any sense.



Seek Home

The *Seek Home* instruction allows you to move the motor until a home sensor is found. The home sensor can be wired to any of the general purpose inputs.

Some applications require the motor to start from a certain position each time you turn on the power, but can't guarantee where it was left at the last power down. The solution is to wire a sensor to one of the Si™ Indexer inputs and place a *Seek Home* command at or near the beginning of the program.



Seek Home will ask for many of the same parameters as the other feed programs: Speed, accel, decel and direction.

You'll also need to tell the Si™ Indexer which input the sensor is wired to and what input condition to look for. The four input conditions are:

High - move until the specified input reaches a high voltage state. This is the default state of an input if nothing is connected to it.

Low - move until the specified input is at a low voltage state.

Rising Edge - move until the signal goes from low to high. This is similar to the high condition, but the difference is important. If you execute a *Seek Home* command to a high input and the load is already on the home sensor (causing the input to be high) then the load will not move. If you choose "rising edge" instead, the Si™ Indexer will move the load to the edge of the home sensor.

If you need the load to be at the exact same position after each Seek Home command, choose Rising Edge or Falling Edge.

Falling Edge - the opposite of rising edge. Si™ Indexer waits for an input voltage to go high, then low.

The Si™ Indexer begins a *Seek Home* command by moving the motor in the direction you have specified. If the home sensor is found, the motor decelerates to a stop, then backs up to the sensor. If a limit is encountered before the home sensor is found, the Si™ Indexer reverses the direction of motion and keeps looking for the home sensor.

You may have noticed a box in the lower right-hand corner of the *Seek Home* dialog box. This tells you how many steps the Si™ Indexer needs to decelerate to a stop. The "Required Clearance" box tells you how much distance you must allow between the limit sensors and any hard stop, based on the speed and decel rate that you've set. If you don't allow enough clearance, the load may crash into something as it decelerates past a limit while seeking home.

The higher the speed, the longer it will take to stop. If the decel rate is increased, then the motor can stop in fewer steps.

Start / Stop Motion

Some applications require a “spindle” that rotates at a constant speed while one or more other axes position a load. Grinding and routing are examples of x-y-spindle applications.

The *Start / Stop Motion* instruction allows the hub to launch an axis into continuous motion. It can also stop a an axis. On some drives (most notably the servos and the 3540i) you can use the *Start / Stop* instruction to change the speed of a motor “on the fly.”

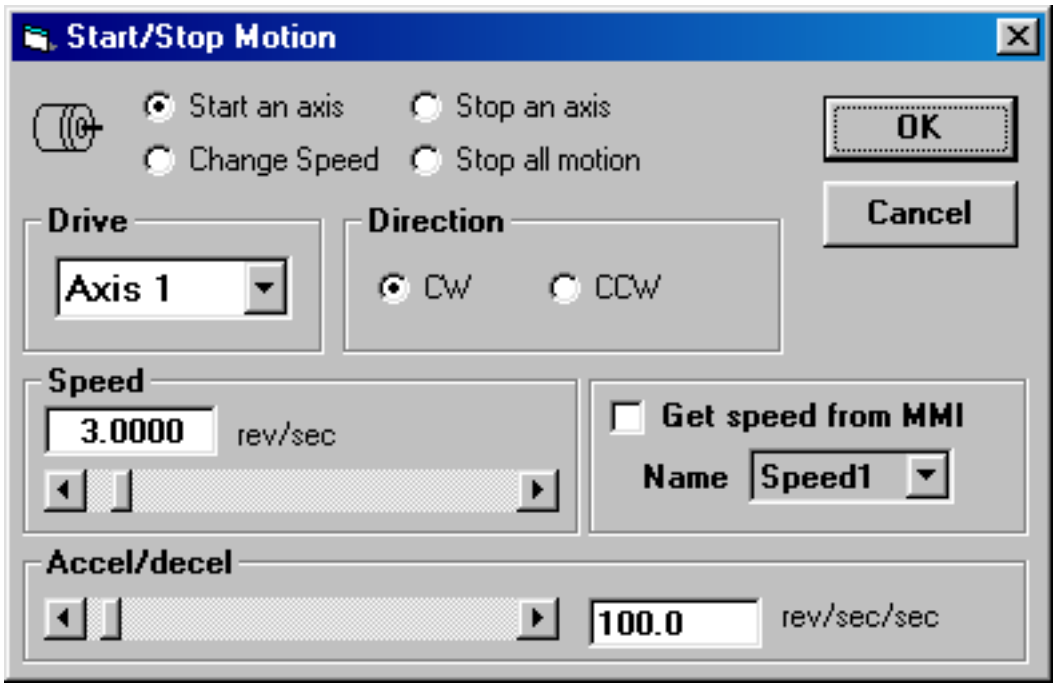
The example program “grinder”, which is installed to the same folder as the *Hub Programmer* software, demonstrates the use of the *Start / Stop* instruction for a spindle application. The MMI offers menu that includes “start the grinding wheel”, “stop the wheel”, “set up” and “move workpiece”.

“Move workpiece” demonstrates the ability to move one axis back and forth (as if an object were passing over a grinding wheel) while the spindle rotates continuously.

“Set up” shows how an operator can use the MMI to adjust the spindle speed, stroke length and number of strokes.

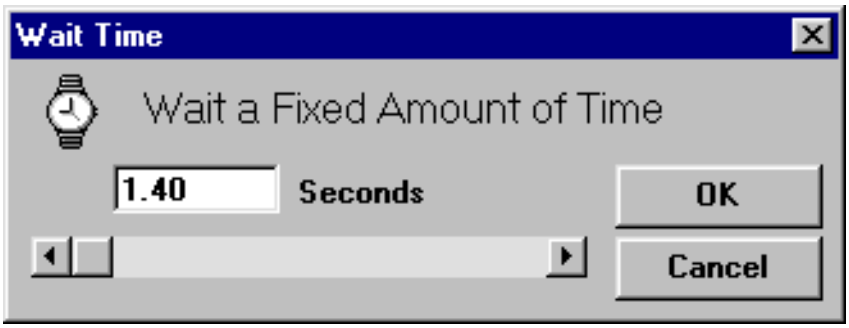
The *Start / Stop* instruction can also stop all motion on all axes. This, when combined with the *If Input* instruction, can provide a means of interrupting a program and stopping all motion. ***However, this is not a fail safe mechanism. It is not a suitable means of preventing injury to people or equipment.***

When you configure a *Start / Stop* instruction to start a move, you specify the speed and acceleration rate. That same accel rate will be used by the *Start / Stop* instruction that stops the motor. If you use *Start / Stop* to interrupt any of the other move instructions, like *Feed to Length*, the deceleration rate will be high and the motor will stop abrupty.



Wait Time

This is the simplest instruction. Just enter an amount of time, and the SiNet™ Hub will pause for that time

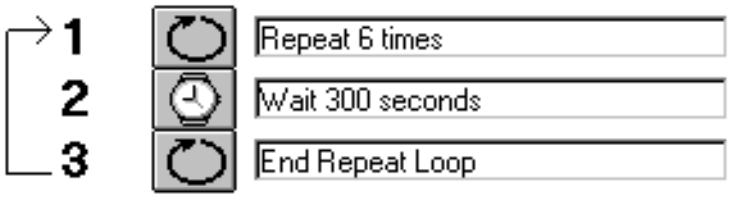


before proceeding to the next line in the program The range is 0.01 to 300 seconds.

What, you want to pause for more than 300 seconds? Did I hear you say 30 minutes? Okay, we can do that. That's the beauty of multiple line programs with a wide range of instructions - you're only limited by your creativity.

You can make the *Wait Time* instruction last longer by placing a repeat loop around it. Now I know we haven't talked about repeat loops yet, so we're going to skip ahead a little here. The first trick is to factor your 3 minute delay into two parts. 30 minutes is 1800 seconds, right? The most we can delay in one *Wait Time* instruction is 300 seconds. Okay, what if we delay for 300 seconds 6 times?

Your program would look like this:



What's the limit? Well, a repeat loop can go 65535 times, so the maximum time you can delay is 65535 x 300 = 19.66 million seconds, or 5461 hours. Not long enough? Try two repeat loops, one inside the other. Now we're pausing for up to 40000 years. Wow!

Wait Input

Rarely does a motion controller operate completely on its own, with no input from the outside world. In most cases, you'll need your program to wait for something to happen before it goes into motion. The *Wait Input* command is used for that. The *Wait Input* command is also the only instruction that allows you to jog the motor using the JOG CW and JOG CCW inputs.

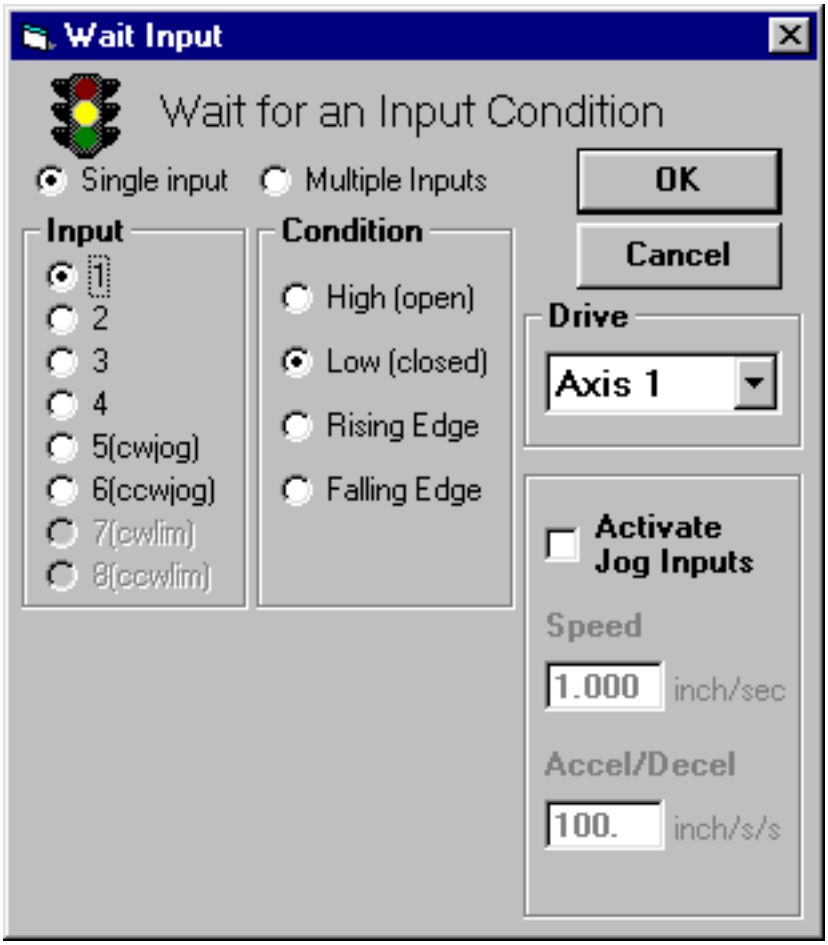
The *Wait Input* instruction has two modes of operation: single input, where it only looks at one input, and multiple inputs where it can examine up to 8 inputs at once.

In Single Input mode, you must specify the input and the voltage condition to expect. The choices are:

High - Wait until the specified input is at a high voltage state. This is the state an input will be in if nothing is connected to it, so be careful if you use this condition. If a wire comes loose, you could end up with undesired motion.

Low - Wait until specified input is at a low voltage state. This happens when the input is conducting current. If you use a momentary contact switch (normally open type), this condition will occur when you press the button.

Rising Edge - Wait until the signal goes from low to high. This is similar to the high condition, but the difference is important. Let's say that your signal into the drive is one that will go high when you want motion to occur. However, the signal remains high after the motion is complete, going low sometime later. If you choose high as your input condition, the program will complete the motion and start again because the input signal is still high when it finishes the first move. If you choose rising edge, the SiNet™ Hub will wait for the input voltage to go low, then high before moving.



Falling Edge - The opposite of rising edge. Hub waits for input voltage to go high, then low.

If you select "Multiple Inputs", the Wait Input instruction will scan up to eight inputs at once to determine if the program should move on to the next instruction. The inputs can be configured as a binary sum (Wait for input 1 low or 3 high) or as a product (Wait for input 1 low *and* 3 high).

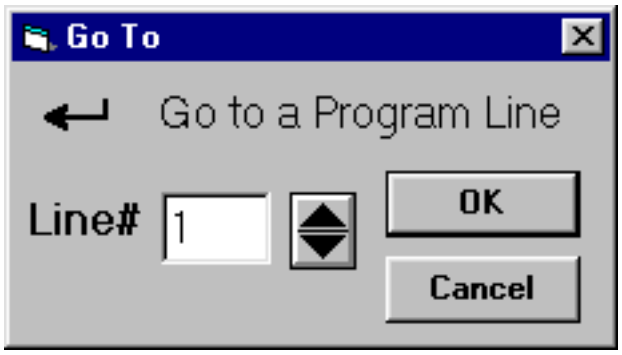
When scanning multiple inputs, all of the inputs must be on the same drive.

The internal inputs of the Hub 444 can be used with the Wait Input instruction.

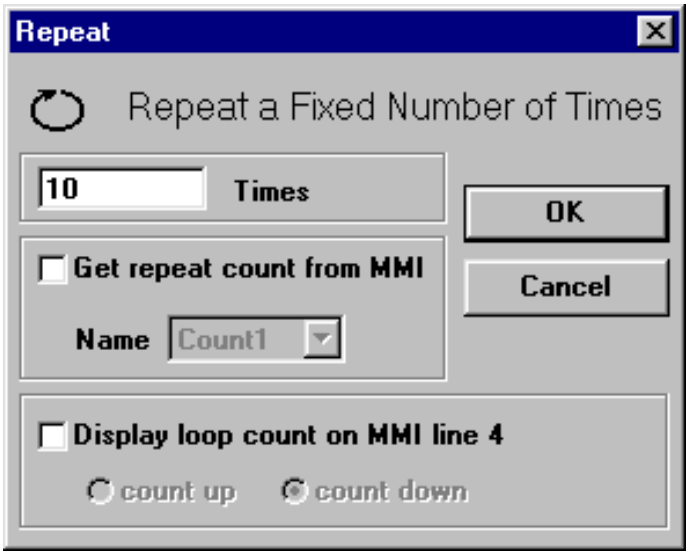
Go To

The *Go To* instruction is used to make the indexer-drive jump to another line in the program. At the least, you'll need to have a *Go To* at the end of your program, to jump back to the beginning.

There is only one parameter to enter in a Go To instruction: the line number you want to jump to. Click on the spin button to increase or decrease the line number.



Repeat/End Repeat



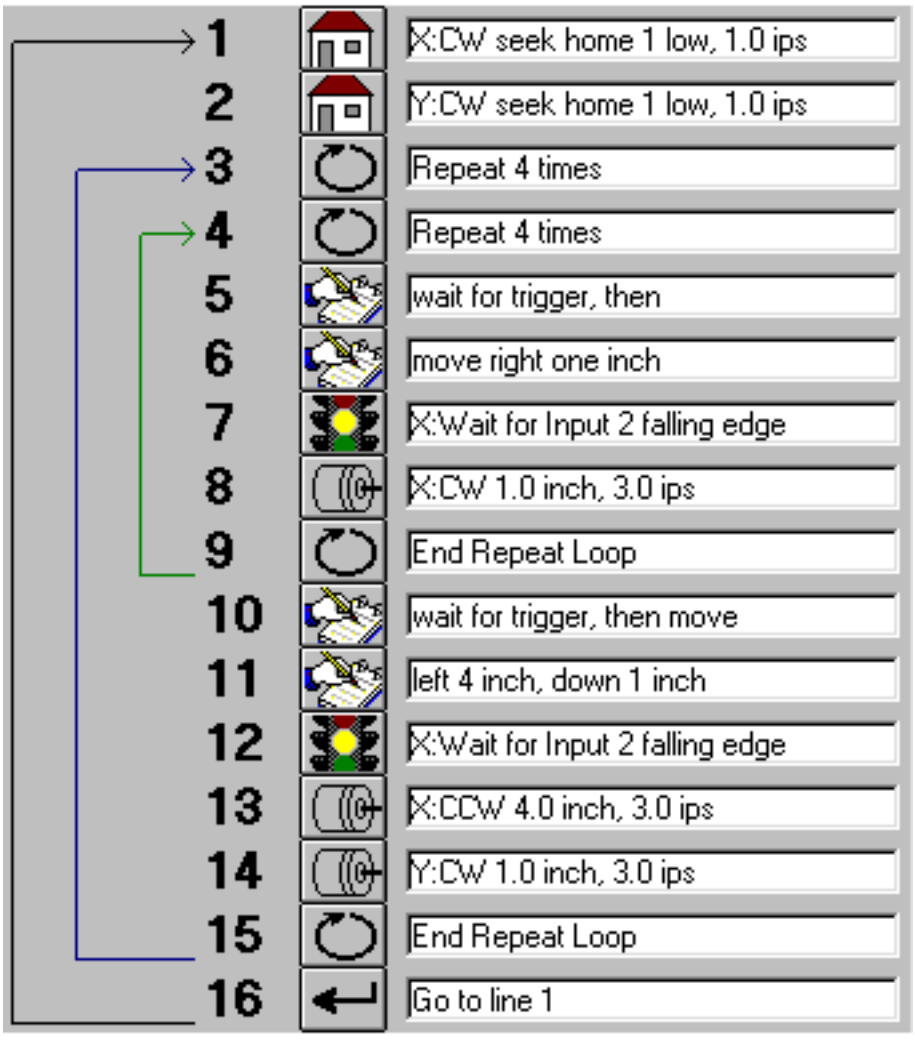
Sometimes you need to do the same thing several times, and you know in advance how many times that is. Repeat loops allow you to repeat the instructions inside the loop up to 65,535 times.

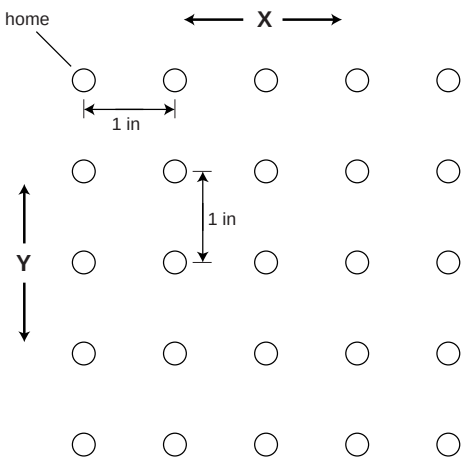
For example, let's say we are dispensing fluids into an array of containers. There are five rows and five columns of containers, each 1 inch away from the next. Each time we receive a trigger command, we want to move to the next position.

Let's also assume that the system is homed to the first position, with the home sensors connected in input 1 on each drive.

We'll use a repeat loop to advance the X axis by 1 inch each time we receive a trigger pulse on input 2 of the X axis drive.

After every fifth container is full, we must return the X axis to the first, 4 inches back, then advance the Y axis one inch.


1 X:CW seek home 1 low, 1.0 ips
2 Y:CW seek home 1 low, 1.0 ips
3 Repeat 4 times
4 Repeat 4 times
5 wait for trigger, then
6 move right one inch
7 X:Wait for Input 2 falling edge
8 X:CW 1.0 inch, 3.0 ips
9 End Repeat Loop
10 wait for trigger, then move
11 left 4 inch, down 1 inch
12 X:Wait for Input 2 falling edge
13 X:CCW 4.0 inch, 3.0 ips
14 Y:CW 1.0 inch, 3.0 ips
15 End Repeat Loop
16 Go to line 1



We'll use a second repeat loop, outside fo the first one, for that. After all 25 containers are full, we'll home the system again.

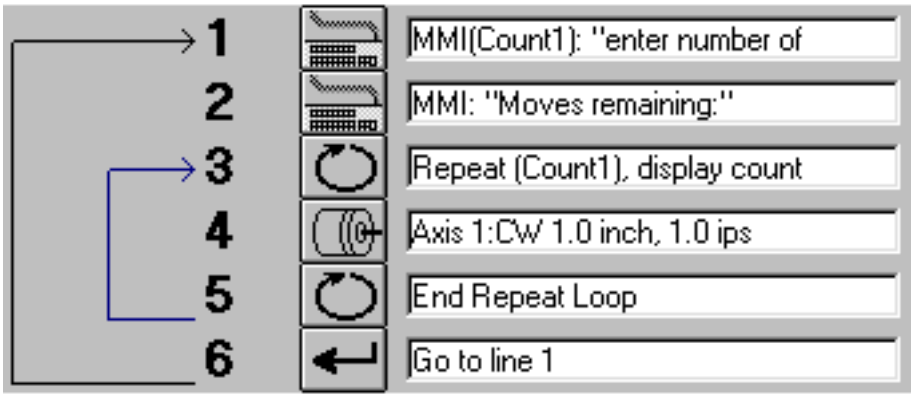
Sometimes you may need to repeat something more than 65,535 times. Let's say your task is to feed material into a cut off knife, and you want to run 100,000 pieces. The best solution is to set up 2 loops, one inside the other. The total number of cycles will be the number of repeats in the two loops multiplied together. 100,000 is 10,000 x 10, so we could set one loop for 10 and the other for 10,000.

The *Repeat* instruction can also use data that was gathered and stored by an *MMI Prompt* instruction as the loop count. Just check the box marked “Get repeat count from MMI” and select a variable from the list.

For example, you could put an *MMI Prompt* in your program to ask for the number of parts to be processed and save that data as Count1. You would then set up the *Repeat* instruction to get the repeat count from the MMI variable Count1.

You can also display the loop count on the MMI as your program runs. You can count up (displaying the number of parts that have been processed, for example) or you can count down (showing the number of parts remaining.)

When the Repeat instruction displays the loop count on the MMI, it uses line 4. You may want to put an MMI Prompt to “display text only” just before the Repeat instruction telling the operator what the count means, as shown below.



Note: If you use an If Input instruction to exit a Repeat Loop, the loop does not automatically reset the next time you enter it. If you exit a loop by using an If Input instruction, you should use a Reset Repeat Loop instruction to reset the loop. Otherwise, the loop count resumes where it left off.

Reset Repeat Loop

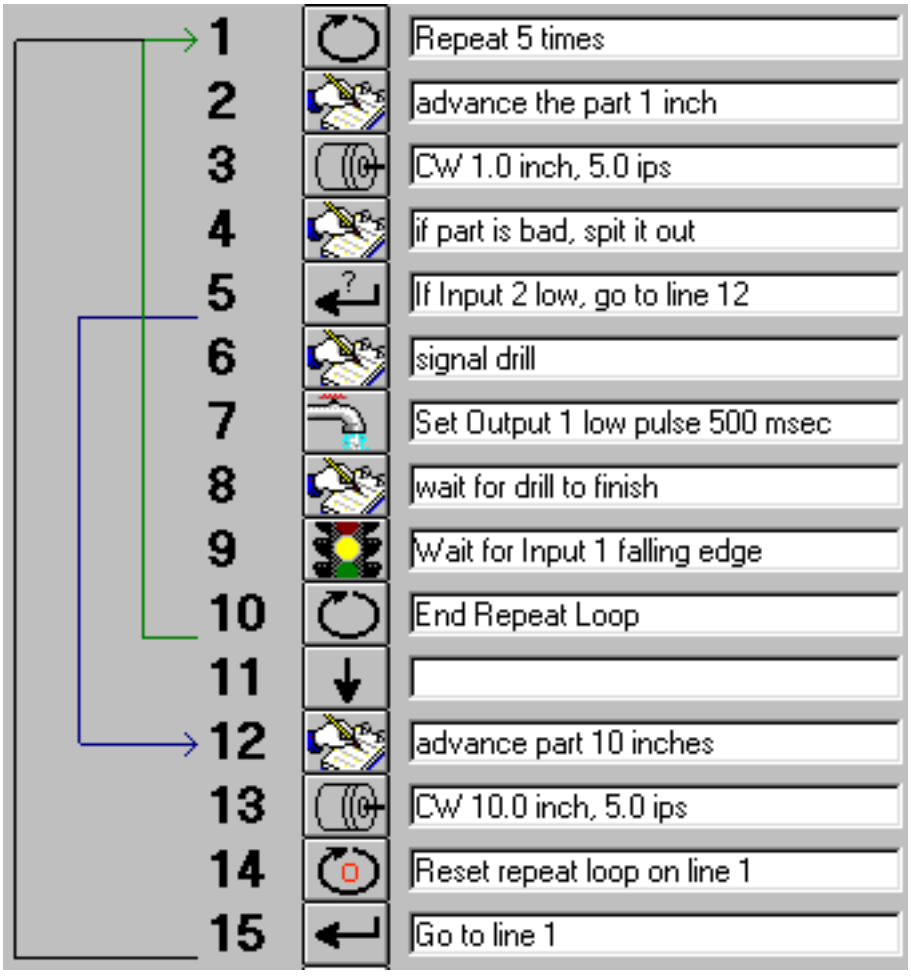
Forces a Repeat Loop to reset its counter if it's been terminated by an If Input or Feed to Sensor with Safety Distance instruction.

Sometimes it is necessary to leave a repeat loop before it is completed. Say, for example, you have a repeat loop that is set up to fill 100 bottles with fluid. If the reservoir runs dry, you want to leave the loop. You can do this by putting an If Input instruction inside the loop, triggered by a fluid sensor. The If Input would branch outside the loop, perhaps to an MMI Prompt telling the machine operator to refill the reservoir.

Now suppose that 60 bottles have been filled, with 40 remaining. If you want the loop to "pick up where it left off", then simply branch back to the beginning of the loop (to the Repeat instruction) after the operator finishes refilling, and the loop will automatically fill the remaining 40 bottles.

On the other hand, what if you are drilling holes in parts, and each part gets five holes. The step motor is used to advance the part by 1 inch for each hole. So, you have a repeat loop with a count of 5. Along comes a bad part, detected after the 3rd hole is drilled. You exit the loop with 2 counts remaining.

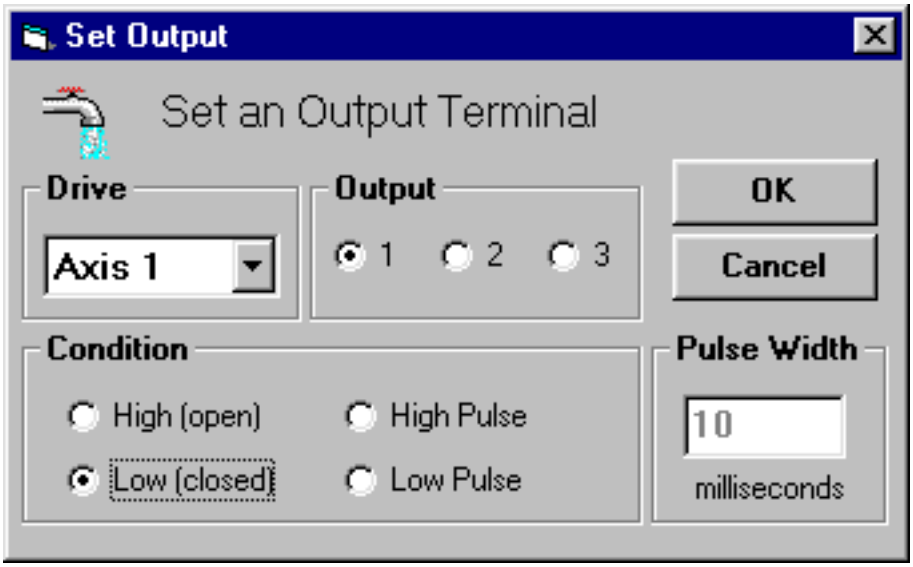
If you simply reenter the loop again, the next part will only get two holes drilled into it. What you really want is for the loop to reset itself. For this, you must use the Reset Repeat Loop instruction, as shown below.


1 Repeat 5 times
2 advance the part 1 inch
3 CW 1.0 inch, 5.0 ips
4 if part is bad, spit it out
5 If Input 2 low, go to line 12
6 signal drill
7 Set Output 1 low pulse 500 msec
8 wait for drill to finish
9 Wait for Input 1 falling edge
10 End Repeat Loop
11
12 advance part 10 inches
13 CW 10.0 inch, 5.0 ips
14 Reset repeat loop on line 1
15 Go to line 1

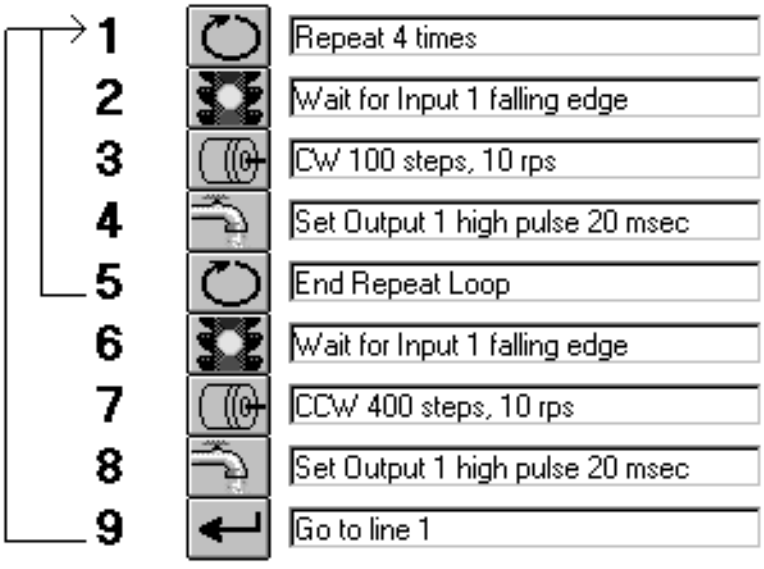
Set Output

Earlier, we discussed the *Wait Input* instruction as a way to make your program wait for external events to happen before proceeding. Sometimes you want the opposite: the SiNet Hub should tell other equipment when to proceed. The *Set Output* command lets you pick one of the three outputs and put a voltage signal on it. For a detailed description of the circuitry and connections, see the section "Wiring Inputs and Outputs" in your drive's hardware manual.

There are four choices of output conditions:
High - Makes the photo transistor open. In circuits where the "-" output pin is grounded, and the "+" pin is pulled up, this causes a high voltage to appear on the "+" pin.
Low - Makes the photo transistor close. In circuits where the "-" output pin is grounded, this causes a low voltage to appear on the "+" pin.
High Pulse - Makes the photo transistor open for a specified amount of time (2 to 500 milliseconds)
Low Pulse - Makes the photo transistor close for a specified amount of time (2 to 500 milliseconds)



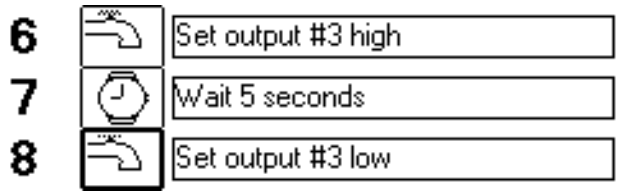
At power-up, all 3 programmable outputs are high (open circuit).


1 Repeat 4 times
2 Wait for Input 1 falling edge
3 CW 100 steps, 10 rps
4 Set Output 1 high pulse 20 msec
5 End Repeat Loop
6 Wait for Input 1 falling edge
7 CCW 400 steps, 10 rps
8 Set Output 1 high pulse 20 msec
9 Go to line 1

For an example of using the *Set Output* instruction in your program, let's consider the example of filling containers. Each time the motor moves to a new position, it should tell the dispenser that it's arrived. We'll do this with a high pulse, but your choice would depend on the kind of signal the dispenser wants to see in order to be activated.

After adding a *Set Output* instruction after each *Feed to Length*, we have the program shown on the left.

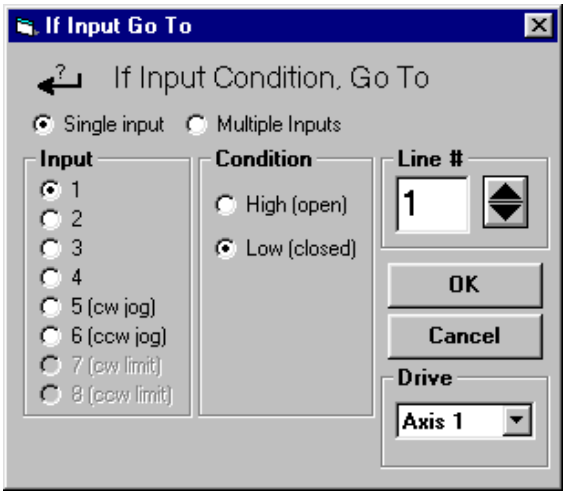
There may be occasions where you want a long pulse. This can be done by combining two *Set Output* commands with a *Wait Time*. The instructions on the left will produce a high pulse of 5 seconds on Output 3.



The internal outputs of the Hub 444 can be used with the Set Output instruction.

If Input Go To

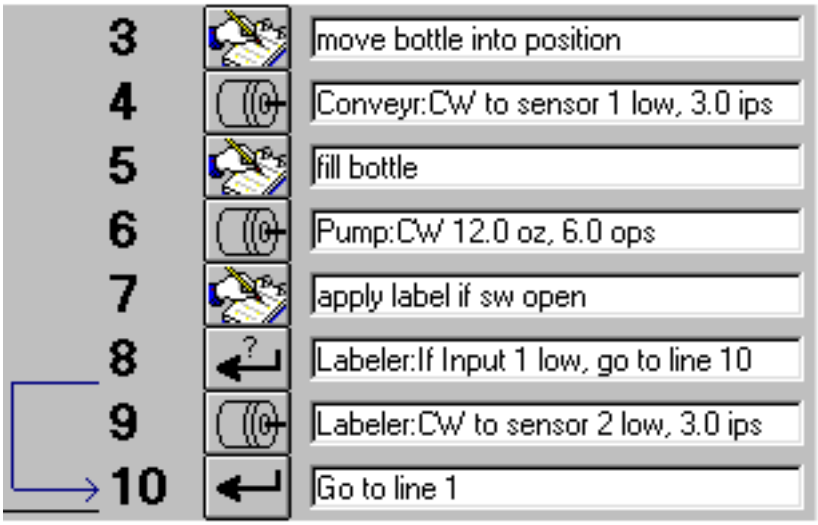
This instruction allows your program to make decisions based on input signals. You can choose “Single Input” or “Multiple Inputs.” **The internal inputs of the Hub 444 can be used with the If Input instruction.**



You’ll need to choose an input terminal and a drive for the instruction to check. You also need to specify what signal condition to look for. Finally, you must set the line number that the instruction will jump to if the input condition occurs.

We included the *If Input* instruction for three reasons.

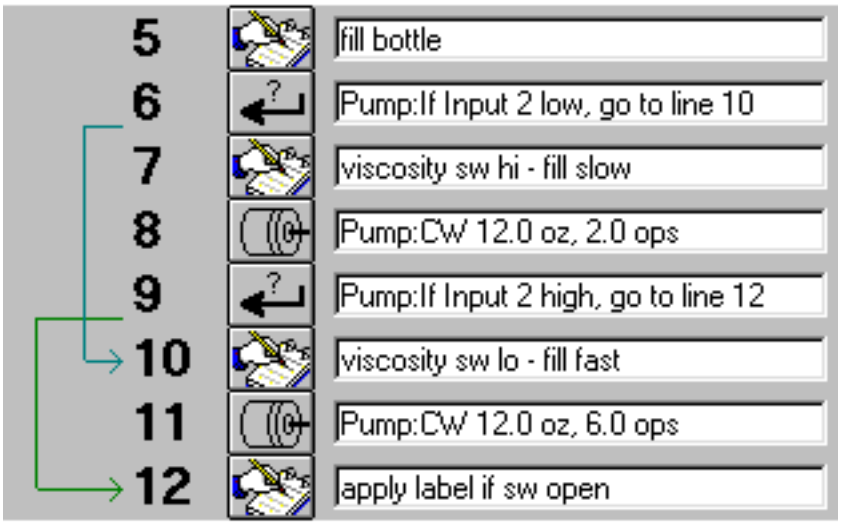
1) It allows you to skip part of your program based on an external condition. For example, let’s say you are building a bottling machine. One axis is dedicated to the conveyor and another to the fill pump, and a third axis to a labeling motor. On some days, the machine is required to put labels on the bottles, but other times the empty bottles already have labels on them.


3 move bottle into position
4 Conveyr:CW to sensor 1 low, 3.0 ips
5 fill bottle
6 Pump:CW 12.0 oz, 6.0 ops
7 apply label if sw open
8 Labeler:If Input 1 low, go to line 10
9 Labeler:CW to sensor 2 low, 3.0 ips
10 Go to line 1

The except at the left shows how to use an If Input instruction to skip over the Labeler move when the switch is closed.

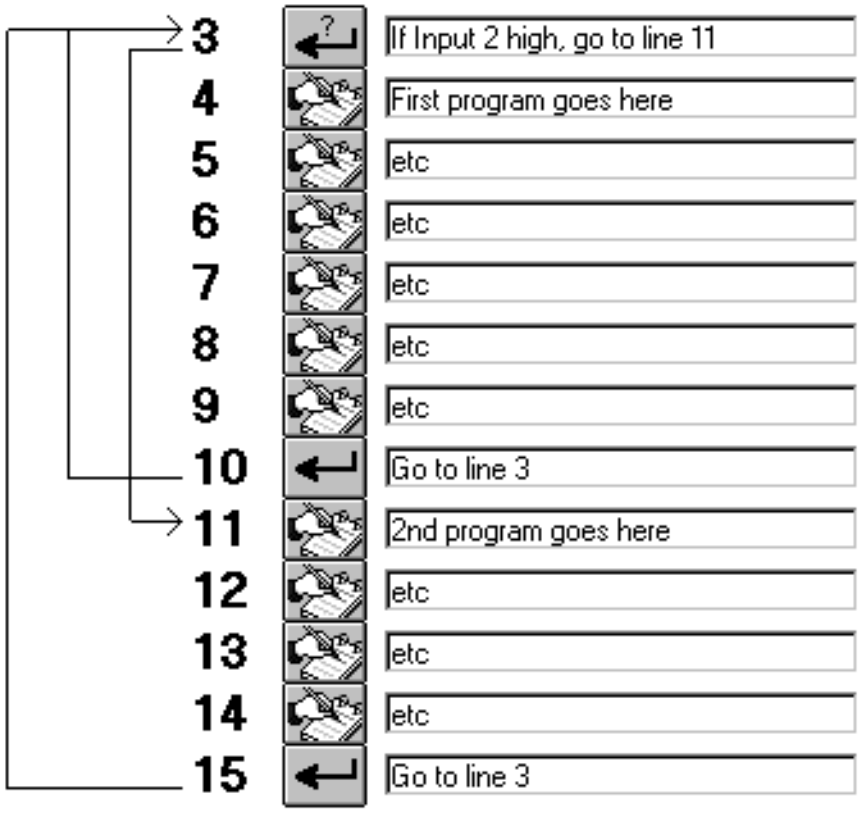
2) The second reason for including the *If Input* instruction is to allow you to change a parameter such as distance or speed based on an input. Consider that last example. If the machine is designed to fill water bottles and shampoo bottles, then the speed of the pump may need to vary depending on the viscosity (thickness) of the fluid.

In the example below, we use a viscosity switch wired to input 2 of the pump motor to decide which pump speed to use.


5 fill bottle
6 Pump:If Input 2 low, go to line 10
7 viscosity sw hi - fill slow
8 Pump:CW 12.0 oz, 2.0 ops
9 Pump:If Input 2 high, go to line 12
10 viscosity sw lo - fill fast
11 Pump:CW 12.0 oz, 6.0 ops
12 apply label if sw open

If the viscosity switch is high, the program executes the move on line 8, filling the bottle at 2 ounces per second. If the switch is low, line 8 is skipped over, and line 11 runs. The bottle is filled at 6 ounces/sec.

3) The final reason we’ve given you the power of *If Input* is so you can have multiple programs within your 200 line program space. Perhaps what you want your system to do is two completely different things depending on an input. Lets say that each of these tasks requires 4 instructions. The next page show what to do.



Depending on the state of Input 2, the program will either execute lines 4 - 9 or lines 11 - 15. Either way, the program ultimately returns to line 3 to check the condition of the switch again.

You can also use the optional Man Machine Interface (MMI) as a decision making input by using the “branch on YES” option of an MMI Prompt instruction. If the operator presses the YES button, the drive will jump to the line you’ve specified in the line number box. If the operator presses NO, the program moves on to the next line.

Note: If you use an If Input instruction to exit a Repeat Loop, the loop does not automatically reset the next time you enter it. If you exit a loop by using an If Input instruction, you should use a Reset Repeat Loop instruction to reset the loop. Otherwise, the loop count resumes where it left off.

If Moving

Because the hub can command a motor to move and then proceed with the program, it is possible to encounter a move again before the first move is complete.

This will only happen if you have selected the “start move and immediately proceed to the next instruction” option in your moves. Even then, it will not occur unless you’ve provided a means for your program to get to another move on the same axis. Un-checking the “Wait for all move to finish” option in a Repeat, Goto or If Input instruction will do the job.

The *If Moving* instruction allows your program to detect that a move is not done and skip over it. This can provide more independence between the axes.

In the sample program “IfMoving”, which is installed to the same folder as the Hub Programmer software, we allow two drives to work independently yet share a single MMI. Each drive executes a move when it’s Input 1 is closed.

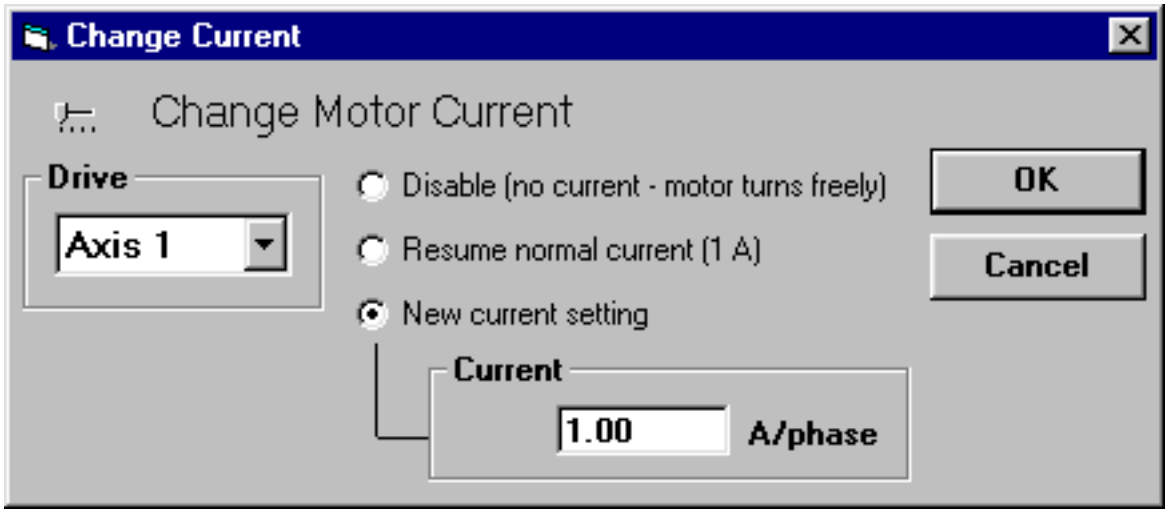
The Hub sits in a loop waiting for input 1 of either drive to go low. For each axis, the hub skips over the move instruction if the input is not low. The If Moving instruction makes it skip over the move if it is already in progress.

Change Current

Motor current is normally set on the main screen of the *SiNet Hub Programmer™* and applies to all program instructions. But what if you want to change the current on command? For example, you may want to temporarily turn off the motor current while the operator makes manual adjustments to a mechanism, or loads a new roll of labels. Overcoming the holding torque of a step motor by hand can be difficult, so it's sometimes best to shut off the current completely.

In other cases, you may want to temporarily increase the motor current to achieve more torque, but are unable to leave it that way all the time without overheating.

The Change Current instruction allows you to turn off the current, to resume the normal current setting, or to specify a new current setting.



Comment

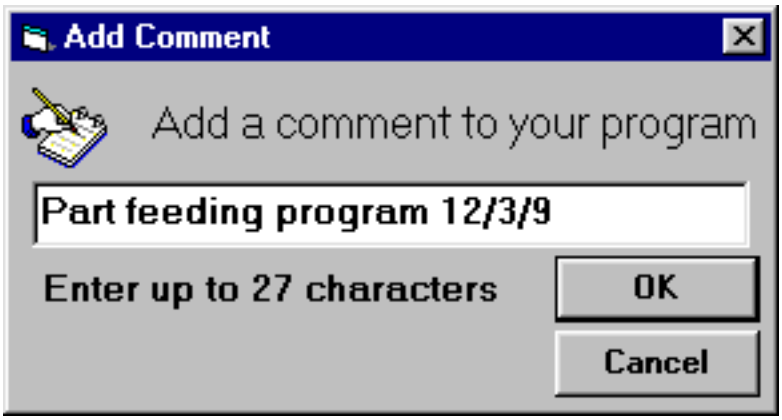
The Comment lets you leave notes in your program. That way if someone else needs to modify your program in the future they'll understand what you've done. The comments can also help you organize the program for yourself.

Whether you save your program to disk or just download it to the drive, the comments stay with it. They do not affect the way your program runs: when the SiNet™ hub executes a program, it skips over the comments.

We suggest that you place a comment on the first line of your program to let future programmers know who wrote the program, when it was written, and what it does. You can also put in comments at other places in your program. Perhaps just before a Feed to Length move, you would want to add the comment "Rotate the part 1/4 turn."

A little time spent now commenting your program might save you a great deal more time later on, when you've forgotten that "Wait for Input 1 low" means "wait photosensor to detect part."

There is no limit to the number of comments your program can have, except that your program cannot exceed 200 lines.



Call and Return

Call and Return instructions are used for subroutines.

Subroutines are useful in many ways. You may find yourself putting the same sequence of instructions in various parts of your program. You could instead put those lines in just one place, maybe at the end of the program, and replace their original locations with Call instructions. That makes a long program shorter. It also makes the program easier for someone else to understand.

If you use the same MMI Prompt in more than one place, you can save space in the string pool by putting the MMI Prompt into a subroutine. For more information about the string pool, see page 22.

Subroutines can also save you time when developing your program. If you decide to make a change in one of those instructions that you keep repeating, if it's in a subroutine you only have to change one line.

To make a subroutine, just place some instructions in your program, then follow them with a Return instruction. To call your sub, just place a Call instruction and enter the line number of the subroutine.

Note: inserting and deleting lines in your program causes some lines to be renumbered. You should check your subroutine calls after you do an insert or delete to make sure everything still makes sense.

Examples of a subroutine, and how to call it, are shown below.